

**SIMATIC S7**

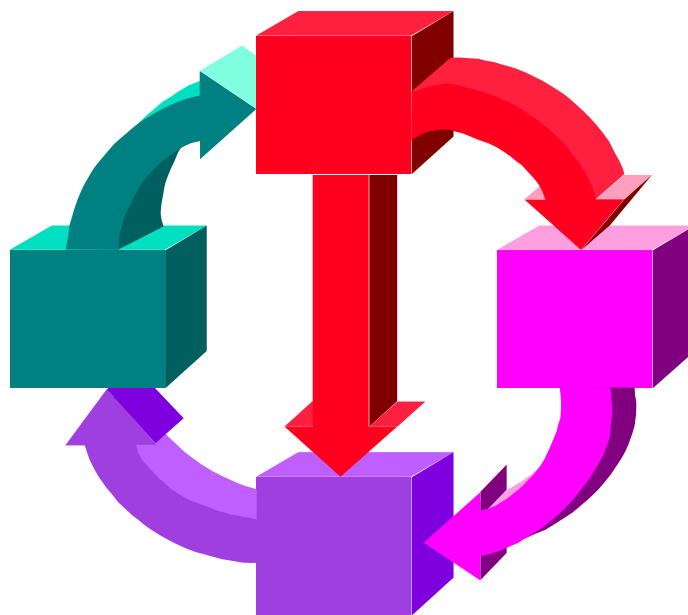
# Programação Avançado

Nome: \_\_\_\_\_

Curso: de \_\_\_\_\_ até \_\_\_\_\_

Instrutor: \_\_\_\_\_

## Instruções Dependentes do Estado Lógico Binário



SIMATIC S7

Siemens AG 1998. All rights reserved.

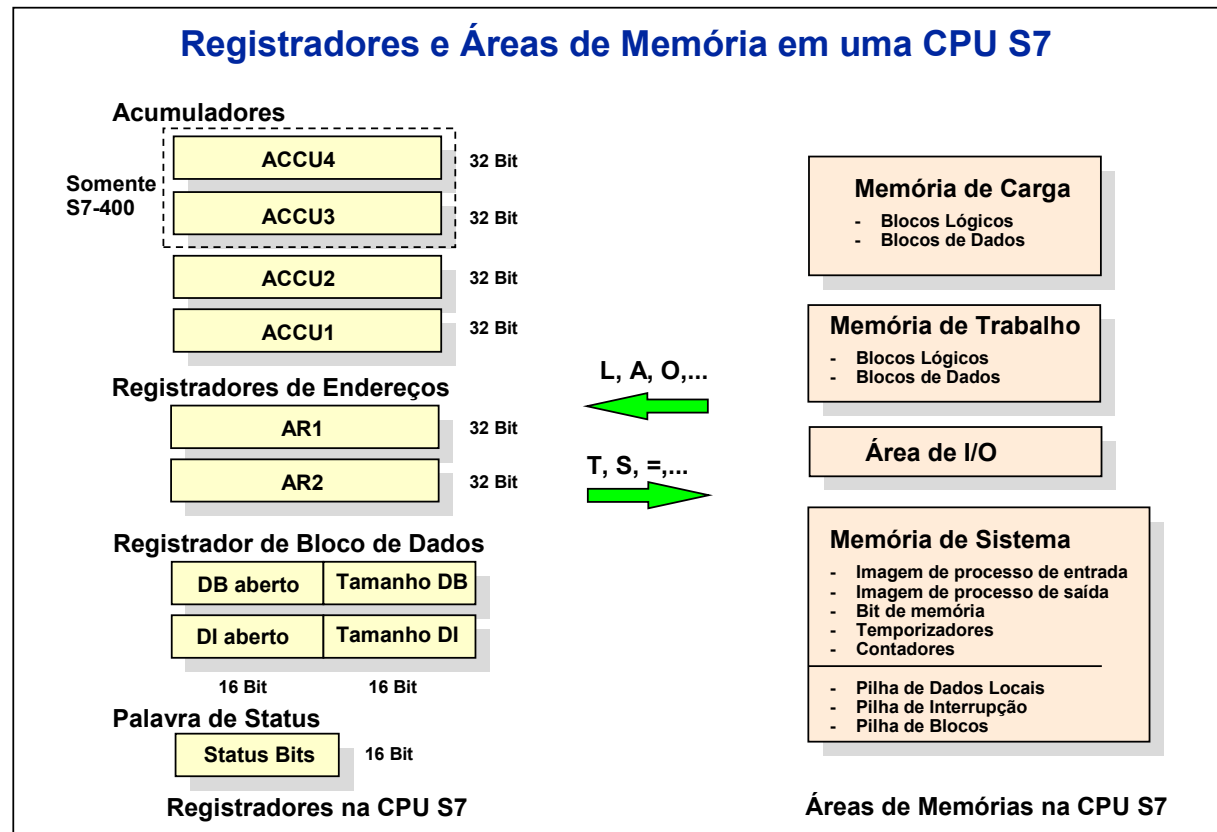
Data: 4/10/2007  
Arquivo: PRO2\_01P.1Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Registradores e Áreas de Memória em uma CPU S7.....	2
Estrutura da Status Word .....	3
Checando os Bits de Status .....	4
Instruções com os Status Bits .....	5
Bit BR e ENO em uma Chamada de Bloco ou Funções Complexas .....	6
Funções de Salto Dependentes do Status Bits .....	7
Funções de Salto Dependentes dos Códigos de Condição .....	8
Programação do Distribuidor de Saltos .....	9
Programação de Instruções de Loop .....	10
Instruções para Fim de Bloco .....	11
Exercício 1.1: Salto depois de uma Subtração .....	12
Exercício 1.2: Salto depois de uma Multiplicação .....	13
Exercício 1.3: Programando um Distribuidor de Saltos .....	14

## Registradores e Áreas de Memória em uma CPU S7



SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.2Conhecimento em Automação  
Training Center

### Visão Geral

Uma CPU-S7 contém vários registradores e áreas de memória para assegurar uma execução eficiente do programa.

**Registradores da CPU** Os registradores da CPU são usados para endereçar ou processar dados. Os dados podem, com ajuda de comandos associados (L, T,...), ser trocados entre áreas de memórias e registradores da CPU:

- **Acumuladores:** Dois (com S7-300) ou quatro (S7-400) acumuladores são usados para instruções byte, word e double word em operações aritméticas, comparações e outras.
- **Registrador de Endereços:** Dois registradores de endereço são usados como ponteiros para o registro de endereçamento indireto.
- **Registrador de Bloco de Dados:** Os registradores de bloco de dados contêm os números dos blocos de dados que estão abertos(ativos). Assim é possível que dois DBs estejam simultaneamente abertos; um DB com ajuda do registrador de DB e o outro como DB instance com o registrador DI.
- **Palavra de Status:** Contém vários bits, que refletem o resultado ou o estado lógico de instruções individuais dentro da execução de programa.

### Áreas de Memória

A memória das CPUs S7 podem ser divididas em quatro áreas:

- A memória de carga é usada para armazenar o programa de usuário sem a função de endereçamento simbólico ou comentário. A memória de carga pode ser RAM ou memória FEPRM.
- A memória de trabalho (RAM integrada) é usada para armazenar a parte pertinente do programa de S7 necessário para sua execução. A execução do programa utiliza exclusivamente partes da memória de trabalho.
- A área de I/O permite o acesso direto das entradas e saídas dos sinais conectados aos módulos ou dispositivos de campo.
- A memória do sistema (RAM) contém áreas, como a tabela de imagem de processo de entradas e saídas, memória de bits, temporizadores e contadores. Adicionalmente, pilha de dados locais, pilha de blocos e a pilha de interrupção.

## Estrutura da Palavra de Status

### Significado dos bits na palavra de status

Bit	Tarefa	Grandeza	Significado
0	/FC	$2^0$	Primeiro Cheque
1	RLO	$2^1$	Resultado Lógico da Operação
2	STA	$2^2$	Status
3	OR	$2^3$	Or (ou)
4	OS	$2^4$	Estouro Armazenado
5	OV	$2^5$	Estouro
6	CC0	$2^6$	Bit de Resultado
7	CC1	$2^7$	Bit de Resultado
8	BR	$2^8$	Resultado Binário
9...15	sem função	$2^9 \dots 2^{10}$	

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.3Conhecimento em Automação  
Training Center

#### Palavra de Status

Os bits individuais da palavra de status, fornecem a informação sobre o resultado ou o estado lógico da instruções executadas como também erros que possam ter surgido.

Você pode integrar o estado do sinal dos status bit's diretamente em seu programa usando lógicas binárias e assim controlar o fluxo do programa.

#### Primeiro Cheque

O bit 0 da palavra de status é chamado de bit de primeiro cheque (/FC= First Check). A condição "0" no bit /FC indica, que uma nova seqüência lógica está começando em seu programa. O barra diagonal na frente da abreviação FC indica que o bit de /FC é negado.

#### Resultado Lógico

O bit 1 da palavra de status é o bit RLO (RLO= "Result of Logic Operation"). É usado como memória temporária em operações de lógica binárias. Uma instrução gerada pelo texto de uma instrução lógica checa, por exemplo, o estado do sinal de um contato e o combina logicamente com o resultado do cheque (bit de estado) de acordo com as regras da lógica booleana com o bit de RLO. O resultado da operação lógica é armazenado de volta no bit de RLO.

#### Bit de Status

O bit de status (bit 2) salva o valor de um bit endereçado. O bit de status sempre mostra, por varredura (A, AN, O,...) ou instruções de escrita (=, S, R,) o estado do bit endereçado.

#### Bit OR

O bit OR é utilizado quando você executa uma operação lógica AND antes de uma OR com a instrução O. O bit OR indica que uma operação lógica AND foi executada anteriormente e recebeu o valor "1", pelo qual o resultado da operação lógica OR está sempre determinado como sendo "1."

#### Bit OV

O bit OV (estouro de capacidade) mostra um erro em uma instrução matemática ou em uma instrução de comparação com números em ponto flutuante. O bit é "1" ou "0" de acordo com o resultado da operação da instrução matemática ou da instrução de comparação realizada.

## Checando os Bits de Status

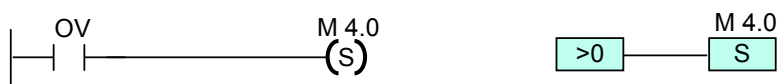
### ❑ Cheque em STL

- A OV      Verifica o estouro de capacidade
- A OS      Verifica o estouro de capacidade memorizado
- A BR      Verifica o bit de memória BR

### ❑ Cheque do Resultado Binário (CC0, CC1)

- A ==0      Resultado igual a 0
- A > 0      Resultado maior que 0
- A <>0      Resultado não igual a 0
- A =<0      Resultado menor que ou igual a 0
- etc.
- A UO      Operação não permitida

### ❑ Cheque em LAD e em FBD



## SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.4



Conhecimento em Automação  
Training Center

### Bit OS

O bit OS (estouro de capacidade memorizada) é estabelecido juntamente com o bit OV. O bit OS permanece memorizado depois de uma nova instrução matemática, quer dizer, não é mudado pelo resultado da próxima instrução matemática.

Assim você tem a oportunidade, até mesmo em um local posterior em seu programa, avaliar um estouro de capacidade de um número ou uma instrução com números Reais inválidos.

O bit OS somente é zerado com os comandos: JOS (salta, se OS = 1), chamadas de bloco e fim de bloco.

### CC1 e CC0

Os bits CC1 e CC0 (códigos de condições) informam sobre os seguintes resultados:

- resultado de uma instrução matemática.
- ou instrução de comparação.
- uma instrução lógica de palavra, ou
- sobre os bits jogados fora por uma instrução de deslocamento.

Os códigos de condição CC1 e CC0 podem ser testados indiretamente por meios das seguintes instruções.

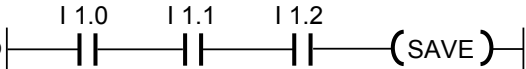
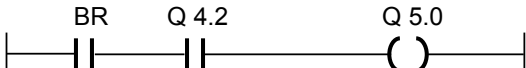
CC1	CC0	Checa, se	
0	0	A ==0	Resultado = 0 (ACCU2 = ACCU1)
1	0	A >0	Resultado > 0 (ACCU2 > ACCU1)
0	1	A <0	Resultado < 0 (ACCU2 < ACCU1)
1	1	A AO	Operação inválida (por ex. divisão por 0).

Adicionalmente, existem funções de salto(Jump), que verificam os códigos de condição e assim permitem que o programa seja estruturável.

### LAD/FBD

Você pode encontrar estas funções representadas em LAD ou FBD, no catálogo de instruções na pasta bits de status.

## Instruções com Bits de Status

Instrução	Significado	Exemplo
SET	Fixa o RLO em "1"	SET //RLO-1-bit de memória = M 0.1
CLR	Fixa RLO em "0"	CLR //RLO-0-bit de memória
NOT	Inverte o RLO	O Manual O Automático NOT; = modo de operação = M0.0
SAVE	Salva o RLO no resultado binário	
A BR	Verifica o resultado binário	

**SIMATIC S7**  
Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.5



Conhecimento em Automação  
Training Center

### L STW/T STW

Também é possível carregar a palavra de estado (status word) inteira e salvá-la para uma verificação posterior (scan).

- L STW Carrega a palavra de status
- T MW 114 Salva na palavra de memória 114

Com a instrução de T STW, a palavra de estado pode, por exemplo, ser carregada com uma palavra de estado previamente salva. Os bits 0, 2, 3, 9 ..15 não são influenciados por esta instrução.

### Mudança de RLO

Com STEP 7 existe um número de instruções para influenciar o RLO.

Com SET você fixa o resultado lógico da operação em "1", com CLR para "0". Paralelo a isto, o bit da status STA também pode ser fixado em "1" ou em "0". Ambas as instruções são independente das condições.

SET e CLR também zeram os bits de estado OR e /FC, isto é, uma nova linha de condições começa em seguida.

A instrução NOT nega ou inverte o resultado lógico da operação.

### Bit BR

O bit BR representa um bit de memória interna, na qual o RLO pode ser salvo antes de uma nova variação do RLO. Isto está desta forma, para que o RLO, novamente esteja disponível para o reinício de uma linha de condições que tenha sido suspensa.

Se você edita um bloco de função ou uma função e quer chamar isto em LAD, você tem que administrar o bit BR. O bit BR corresponde a saída (ENO) para os blocos em LAD.

### Fixar e Zerar o Bit BR

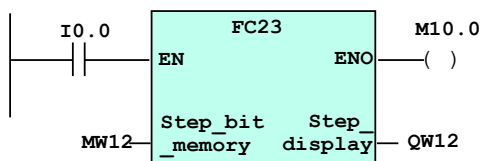
Com a instrução SAVE você salva o RLO no registrador de resultado binário (BR). A instrução SAVE transmite o estado do RLO para o bit BR.

SAVE é executada independente de qualquer condição e não afeta nenhum dos demais bits de status.

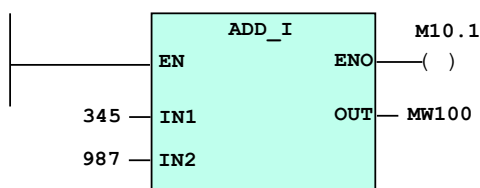
## Bit BR e ENO em uma Chamada de Bloco ou Função Complexa

### LAD

Network 1: Programa Cíclico



Network 2: ???



### STL

Network 1: Programa Cíclico

```

A      I      0.0
JNB    _001
CALL   FC     23

      Step_bit_memory      :=MW12
      Step_display         :=QW12
_001: A      BR
      =      M      10.0

```

Network 2: ???

```

L      345
L      987
+I
T      MW     100
AN     OV
SAVE
CLR
A      BR
      =      M      10.1

```

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.6Conhecimento em Automação  
Training Center

**EN = habilita entrada** O usuário pode modificar a chamada com ajuda da entrada EN(enable input) que existe em toda caixa de chamada de bloco ou uma função complexa em Diagrama de Contatos (corresponde à chamada condicional em STEP 5).

- Se EN não estiver ativada (p.e. o sinal estiver em estado "0"), então a caixa não executará sua função. A saída de habilitação ENO (enable output), conseqüentemente, também não será ativada.
- Se EN estiver ativada (p.e. o sinal estiver em estado lógico "1"), então a função da caixa será executada.

### ENO = habilita saída

O bloco chamado ou a função complexa pode ser sinalizado, com ajuda da habilitação de saída ENO, se não for executado ou executado sem erro. O bit-BR da palavra de status está disponível ao usuário para se salvar erros. O bit-BR é automaticamente fixado em 1 na partida de uma classe de prioridade. Subseqüentemente o bit-BR só é modificado pelos blocos, não pelo sistema.

Se um erro surgir durante o processamento, o usuário pode "salvar" este estado de erro através do reset do bit-BR. Depois do processamento de uma caixa em LAD/FBD, o estado do bit-BR é copiado para o parâmetro de saída ENO.

Então, um mecanismo de análise de estado de erro está disponível no STEP 7. Desta forma, por exemplo, um bloco chamado pode informar a chamada de blocos se o processamento foi executado livre de erro ou não.

### Nota

O parâmetro de EN não é um parâmetro de entrada verdadeiro. Se este for criado, então duas instruções uma com um salto condicional para um rótulo (Label) e o rótulo recuado são gerados automaticamente.

Da mesma maneira, ENO não é um parâmetro de saída verdadeiro. Se ENO é determinado, então são geradas duas instruções para copiar o bit BR no parâmetro de saída atual, automaticamente.

## Funções de salto (Jump) dependentes dos Bits de Status

JU Label <sup>1)</sup>	Salto Incondicional
JC Label <sup>1)</sup>	salta se o bit "RLO" = 1
JCN Label <sup>1)</sup>	salto se o bit "RLO" = 0
JCB Label <sup>1)</sup>	salto se o bit "RLO" = 1 e salva RLO
JNB Label <sup>1)</sup>	salto se o bit "RLO" = 0 e salva RLO
JB I Label <sup>1)</sup>	salto se o bit "BR" = 1
JBN I Label <sup>1)</sup>	salto se o bit "BR" = 0
JO Label <sup>1)</sup>	salto se o bit "OV" na palavra de status = 1
JOS Label <sup>1)</sup>	salto se o bit "OS" na palavra de status = 1

<sup>1)</sup> Rótulo (Label) pode ser constituído de até 4 dígitos alfanuméricos

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.7



Conhecimento em Automação  
Training Center

### Funções de Salto (Jump)

Com esta função de controle lógico você pode interromper o processamento linear de um programa e continuar em um outro local dentro do bloco. Uma ramificação de programa pode ser executada independentemente de uma condição ou então somente quando uma determinada condição for satisfeita.

### Salto Incondicional

A função de salto JU é sempre executada, isto é, executa o salto independentemente de qualquer condição. JU interrompe o processamento linear de um programa e reinicia no rótulo (Label) da instrução de salto (Jump). JU não afeta os bits de status.

### Funções de Salto com RLO e BR

Uma ramificação de programa pode ocorrer dependente do estado do bit RLO e bit BR. Adicionalmente, existe a possibilidade de durante a verificação do bit RLO salva-lo ao mesmo tempo no bit BR.

As funções de salto dependentes do RLO (JC, JCN) são acionadas, não somente para condições atendidas mas também para condições não atendidas, os bits de status STA e RLO vão para "1" e os bits OR e /FC vão para "0".

As funções de salto com "RLO memorizado" (JCB, JNB) salva em todos os casos o estado do bit de RLO no bit BR. Os bits restantes STA, RLO, OR e /FC são de outra maneira tratados do mesmo modo que aquelas funções de salto (jump) que não salva o RLO.

As funções de salto (jump) (JBI, JNBI) dependentes do bit BR determinam os bits da STA, não somente para condições atendidas mas também para condições não atendidas, o bit de estado STA vai para "1" e os bits OR and /FC vão para "0". Os bits RLO e BR permanecem inalterados.



## Funções de Salto dependentes dos Códigos de Condição

JZ Label <sup>1)</sup>	Salta se na palavra de status o bit "CC1"=0 e "CC0"=0 (Resultado = 0)
JN Label <sup>1)</sup>	Salta se na palavra de status o bit "CC1" não for igual a "CC0" (Resultado <> 0)
JP Label <sup>1)</sup>	Salta se na palavra de status o bit "CC1"=1 e "CC0"=0 (Resultado > 0)
JM Label <sup>1)</sup>	Salta se na palavra de status o bit "CC1"=0 e "CC0"=1 (Resultado < 0)
JPZ Label <sup>1)</sup>	Combina os saltos JZ e JP (Resultado >= 0)
JMZ Label <sup>1)</sup>	Combina os saltos JM e JZ (Resultado <= 0)
JUO Label <sup>1)</sup>	Salta se: número real inválido "desordenado" ou divisão por zero

<sup>1)</sup> Rótulo (Label) pode ser constituído de até 4 dígitos alfanuméricos

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.8



Conhecimento em Automação  
Training Center

### Funções de salto com OV e OS

Os saltos JO e JOS são executados se um estouro tiver ocorrido. Em uma rotina de cálculo com diversas instruções executadas sucessivamente, a avaliação do bit OV deve ser realizada após cada função matemática. Uma instrução matemática que gera um estouro, cujo resultado mascara a faixa de números permitidos, zera o bit OV.

Para permitir avaliar um estouro de faixa de números permissíveis no fim de uma rotina de cálculo, checa-se o bit OS. O bit OS somente é zerado com chamada de blocos e fim de bloco bem como com salto JOS.

Os bits restantes na palavra de status não são alterados com as funções de salto JO e JOS.

### Funções de salto com CC0 e CC1

Uma função de programa pode ser dependente dos bits de status CC0 e CC1.

Deste modo, você pode, por exemplo, checar se o resultado de um cálculo é positivo, negativo ou zero.

As funções de salto dependem dos bits de status CC0 e CC1 e não alteram qualquer bit de status. O resultado da operação lógica é monitorável por saltos e pode ser utilizado por conseqüentes operações lógicas no programa do usuário (não alterado pelo /FC).

### Exemplo

Subtração de dois inteiros com subsequente avaliação:

L MW2

L MW8

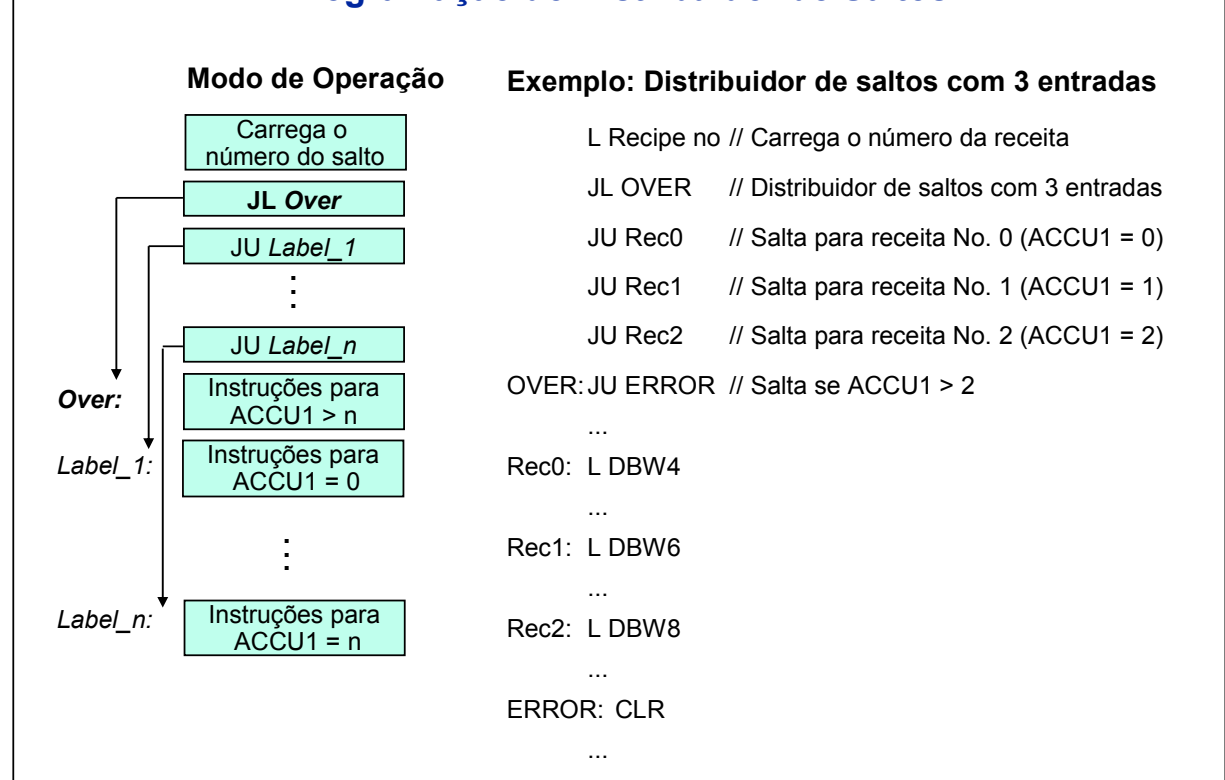
-I

**JZ ZERO** // ocorre o salto se o resultado for igual a "0"

// Instruções, se o resultado for diferente de "0"

**ZERO:** NOP 0 // Instruções p/reação quando resultado for igual "0"

## Programação do Distribuidor de Saltos



SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.9Conhecimento em Automação  
Training Center

**Distribuidor de saltos** O distribuidor de saltos JL permite o salto meta para uma seção do programa dependendo de um número de salto. A instrução JL trabalha junto com uma lista de funções de salto JU.

Esta lista segue imediatamente após JL e pode incluir no máximo de 255 entradas. Com JL existe um rótulo de salto que aponta para o fim da lista, isto é, a primeira instrução após o fim da lista.

Somente instruções JU podem ser utilizadas entre a JL <Label> e o <Label>: <instrução>. Se "0" estiver guardado no ACCU1-L-L, a primeira instrução de salto é executada, com "1" a segunda, etc. Se o número for maior do que o tamanho da lista, JL salta para o fim da lista.

A instrução JL é executada apesar de quaisquer condições e desta forma não altera os bits de status.

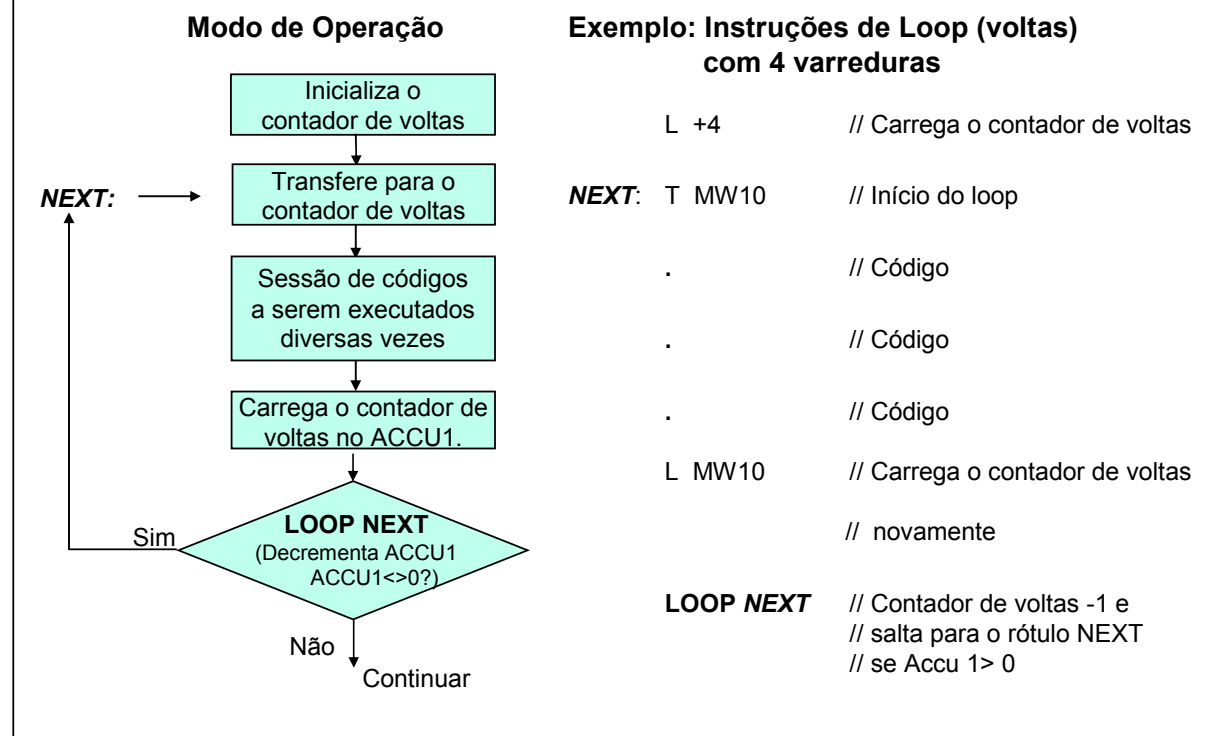
### Nota

Os saltos podem tomar lugar dentro do comprimento total do bloco (mesmo além dos limites do network). Por esta razão, os nomes dos rótulos dos saltos dentro de um bloco devem ser únicos.

Somente é possível saltar dentro do bloco de programa, desde que a distância de salto possa somente ser determinada para os saltos por meio dos rótulos dos saltos. O comprimento do rótulo de salto é limitado a quatro caracteres alfanuméricos, pelo qual o primeiro caractere deve ser uma letra. É feita distinção entre letras maiúsculas e minúsculas no rótulo dos saltos.

Uma instrução deve sempre ser colocada depois do rótulo do salto separada por ":". A máxima distância de salto é -32768 ou + 32767 palavras de código de programa. O máximo número atual de instruções que você pode sobre saltar depende do misto de instruções utilizadas em seu programa (instruções de uma, duas ou três palavras).

## Programação de Instruções de Loop (voltas)



SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.10Conhecimento em Automação  
Training Center

### Instrução de Loop

A instrução de volta LOOP permite uma programação simplificada de loop.

Para a programação da instrução de loop o número desejado de execuções a serem realizadas é carregado no ACCU1-L. LOOP interpreta a palavra da direita do ACCU 1 como um número inteiro não sinalizado de 16 bits na faixa de valores entre 0 a 65535.

Com cada execução da instrução LOOP, o valor em ACCU1-L é decrementado de um. Subseqüentemente, o valor é comparado com zero. Se o valor é diferente de zero, um salto toma lugar para o rótulo de salto designado na instrução LOOP. Se o valor é igual a zero, não há salto e a próxima instrução, após a instrução LOOP, será executada.

### Nota

O contador de voltas não deve ser inicializado com "0", porque isto irá causar um loop a ser executado 65535 vezes.

## Instruções de Fim de Bloco

- ☐ **BE**                      **Fim de Bloco**
  
- ☐ **BEU**                    **Fim de Bloco Incondicional (dentro de um bloco)**
  
- ☐ **BEC**                    **Fim de Bloco Condicional (dependente do RLO)**
  
- (RET)                    **mostrado em LAD**
  
- RET                    **mostrado em FBD**

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.11Conhecimento em Automação  
Training Center

### Funções de Fim de Bloco

Você pode terminar o processamento de blocos com BEC dependendo do resultado lógico operacional ou com BEU ou BE independentemente das condições.

#### BE

A instrução BE termina o processamento do programa no bloco de programa corrente. BE é sempre a última instrução do bloco. Esta é automaticamente criada pelo STEP 7 quando um bloco é salvo. Desta forma, esta não tem que ser introduzida separadamente.

O sistema operacional diversifica suas atividades de chamada de blocos e recomeça o processamento de programas pela primeira instrução após a chamada do programa. A área de dados locais reservada pelo bloco é habilitada uma vez de novo.

#### BEU

A instrução BEU termina o processamento do programa do bloco de programa corrente da mesma forma que BE.

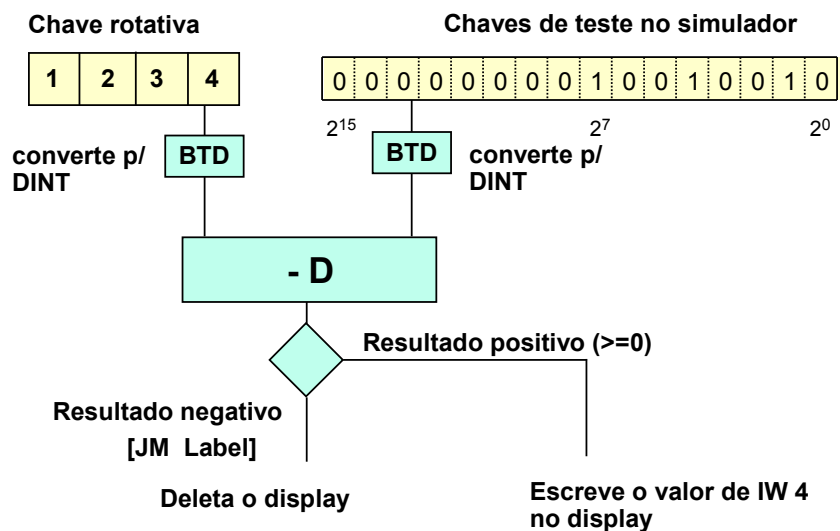
Diferentemente da instrução BE, você pode programar BEU repetidas vezes dentro de um bloco. A secção de programa após a instrução BEU somente será processada se houver um salto para esta parte do programa.

#### BEC

O bloco é finalizado dependendo do valor do bit RLO. Se RLO = 1, o processamento do programa é finalizado no bloco corrente e recomeça no próximo bloco chamado com a primeira instrução após a chamada de programa.

A área de dados locais reservada pelo bloco é habilitada uma vez de novo. Se RLO = 0, a instrução BEC não é executada. A CPU fixa o RLO em "1" e processa a instrução seguinte a BEC.

## Exercício 1.1: Salto após uma Subtração



Chave rotativa: IW4 (IW2, mod. 32 bit)  
 Chaves de teste: IW0 (IW0, mod. 32 bit)  
 Display: QW12 (QW6, mod. 32 bit)

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
 Arquivo: PRO2\_01P.12



Conhecimento em Automação  
 Training Center

### Vista Geral

Com funções de salto, o processamento linear de um programa pode ser interrompido e continuar em outro local. Um salto pode ser realizado, particularmente, em função de condições ou resultados.

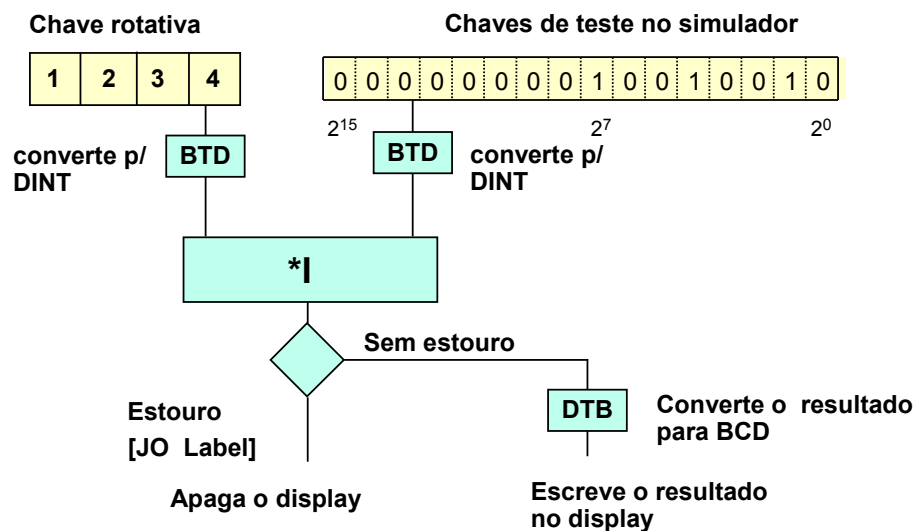
**Objetivo do Exercício** Programação de uma função de salto, esta é executada dependendo do resultado da subtração.

### Enunciado

Gerar um projeto PRO2 e subsequentemente uma pasta de programa S7 com o nome EXERCÍCIO e criar um FC 11 com a seguinte funcionalidade:

1. Carregar a palavra de entrada da chave rotativa e das chaves de teste como valor codificado em BCD nos acumuladores.
2. Subsequentemente executar uma conversão dos valores para DINT. Para a conversão usar o comando BTB (BCD\_TO\_DINT). Este comando assegura que os valores lidos sejam interpretados como um número decimal positivo de quatro dígitos.
3. Então subtrair o valor gerado pelas chaves de teste do valor gerado pela chave rotativa.
4. Executar, dependendo do resultado, as seguintes ações:  
Resultado < 0: apaga o display do simulador, isto é, transfere "0" para o display.  
Resultado >= 0: escreve o valor codificado BCD da chave rotativa no display.  
Notas: usar o comando de salto "JM [Label]" para o caso distinto. Para mascarar erros de conversão durante o ajuste dos números, programar o OB 121 com uma instrução: NOP 0.
5. Chamar o FC 11 no OB 1 e transferir os blocos (OB 1, OB 121 e FC 11) na CPU S7.
6. Teste seu programa.

## Exercício 1.2: Salto após uma Multiplicação



Chave rotativa: IW4 (IW2, mod. 32 bit)  
 Chaves de teste: IW0 (IW0, mod. 32 bit)  
 Display: QW12 (QW6, mod. 32 bit)

SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
 Arquivo: PRO2\_01P.13



Conhecimento em Automação  
 Training Center

**Objetivo do Exercício** Programação de uma função de salto, esta é executada dependendo do resultado da multiplicação.

### Enunciado

Criar um FC 12 com a seguinte funcionalidade:

1. Carregar a palavra de entrada da chave rotativa e das chaves de teste como valor codificado em BCD (sem sinal) nos acumuladores.
2. Subseqüentemente executar uma conversão dos valores para DINT. Para a conversão usar o comando BTB (BCD\_TO\_DINT). Este comando assegura que os valores lidos sejam interpretados como um número decimal positivo de quatro dígitos.
3. Então execute uma multiplicação em 16 bits.
4. Cheque seu resultados dos cálculos para “Estouro” e execute as seguintes ações:

Estouro: apague o display

Sem Estouro: execute a conversão do resultado em um número BCD positivo correspondente e mostre o resultado (pelo menos com os últimos quatro menos significativos) no display.

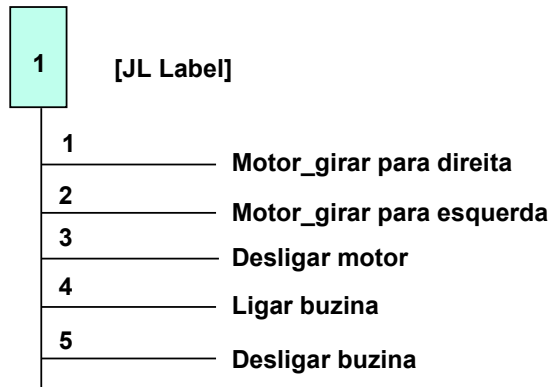
Notas: use o comando de salto "JO [Label]" para testar o “Estouro”. Para mascarar erros de conversão durante o ajuste dos números, programar o OB 121 com uma instrução: NOP 0.

5. Chamar o FC 11 no OB 1 e transferir os blocos (OB 1, OB 121 e FC 11) na CPU S7.
6. Teste seu programa.

## Exercício 1.3: Programando um Distribuidor de Saltos

Função:

Chave rotativa



Label: Saltar via salto para lista

Endereços:	S7-300 (16-Bit)	S7-300 (32-Bit)
	I0.0	I0.0
	Q8.0	Q4.0
Motor_direita:	Q20.5	Q8.5
Motor_esquerda:	Q20.6	Q8.6
Buzina:	Q20.7	Q8.7

SIMATIC S7

Siemens AG 1998. All rights reserved.

Data: 4/10/2007  
Arquivo: PRO2\_01P.14



Conhecimento em Automação  
Training Center

**Objetivo do Exercício** Familiarizar você com o uso de Salto para Lista.

### Enunciado

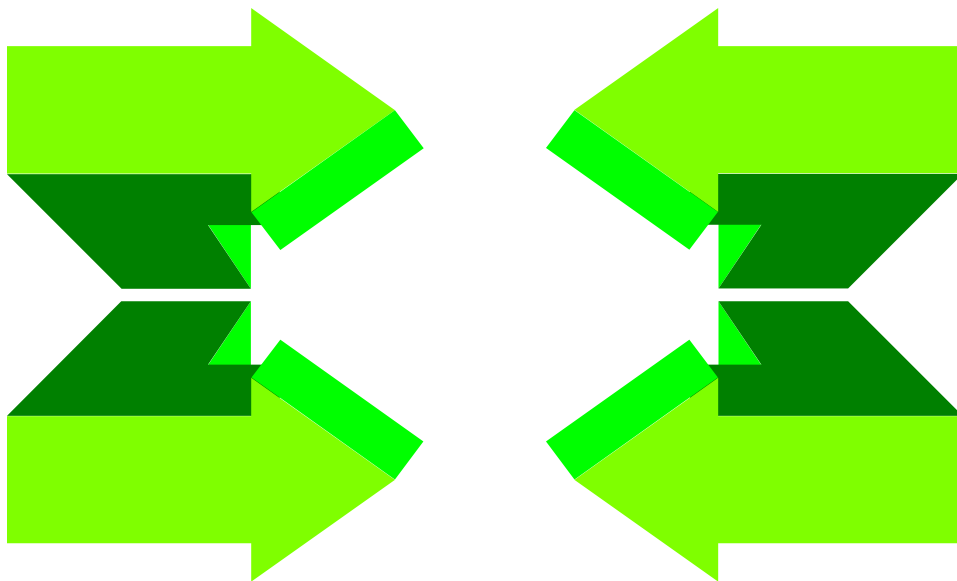
Criar um FC 13 com a seguinte funcionalidade:

- Um número de 1 a 5 pode ser passado via o parâmetro de entrada "Seleção" com o tipo de dado INT.
- Dependendo do número passado, as seguintes ações são executadas:
  - 1: O motor gira para direita
  - 2: O motor gira para esquerda.
  - 3: O motor para.
  - 4: A buzina é ligada.
  - 5: A buzina é desligada.
- Todos os outros números são interpretados como erros, isto é, o "parâmetro de saída" ENO é fixado para FALSO.

### Procedimento

1. Criar um FC 13 com a funcionalidade descrita acima.  
Na implementação do salto para lista note que somente saltos absolutos podem ser utilizados.
2. Chamar FC 13 no OB 1 em dependência do I 0.0.  
O valor do parâmetro de entrada "Seleção" será passado ou ajustado com ajuda da chave rotativa no simulador e ser aceito com um valor de nível positivo no I 0.0.
4. No caso de um erro, isto é, o valor da "Seleção" ser maior que 5 ou menor que 1, a saída Q 8.0 será zerada no OB 1 via o parâmetro ENO.
5. Transferir o OB 1 e o FC 13 e testar o programa.

## Funções com Acumuladores



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.1Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Visão geral das funções com acumuladores .....	2
A instrução TAK (Troca ACCU1 com ACCU2) .....	3
As instruções PUSH e POP .....	4
As instruções ENT e LEAVE (só S7-400) .....	5
Instruções aritméticas .....	6
Instruções lógicas com palavras .....	7
Instruções de troca para ACCU1 .....	8
Instruções incrementais para ACCU1 .....	9
Formando complemento de um .....	10
Negando números (Complemento de dois) .....	11
Instrução de rotação 32 Bits via Bit CC1 .....	12
Exercício 2.1: Cálculo de potenciação .....	13
Exercício 2.2: Troca de dados no ACCU1 .....	14
Exercício 2.3: Formando complementos .....	15



## Visão Geral das Funções com Acumuladores

### ❑ Instruções que modificam vários Acumuladores

- TAK: Troca de conteúdo entre ACCU1 e ACCU2
- PUSH: Deslocando o conteúdo dos ACCU's para cima
- POP: Deslocando o conteúdo dos ACCU's para baixo
- ENT: Deslocando o conteúdo dos ACCU's para cima, sem ACCU1
- LEAVE: Deslocando o conteúdo dos ACCU's para baixo, sem ACCU2
- Instruções aritméticas e instruções lógicas com palavras

### ❑ Instruções que modificam somente o ACCU1

- INC: Incrementa o conteúdo do ACCU 1-L-L
- DEC: Decrementa o conteúdo do ACCU 1-L-L
- CAW: Inverte a ordem dos Bytes no ACCU1-L (16 Bit)
- CAD: Inverte a ordem dos Bytes no ACCU1 (32 Bit)
- INVI, INVD: Formando complemento de um
- NEGI, NEGD, NEGR: Formando complemento de dois (Negação)
- RLDA, RRDA: Rotacionando o conteúdo do ACCU1 para direita ou esquerda via código de condição CC1

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.2



Conhecimento em Automação  
Training Center

### Visão Geral

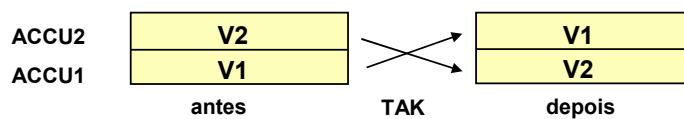
As funções de acumulador transmitem valores entre os acumuladores ou troca de bytes no acumulador 1. A execução de funções puras de acumulador é independente do resultado lógico da operação ou dos bits de status.

Igualmente, nem o resultado de operação de lógica nem os bits de status são afetados pela execução.

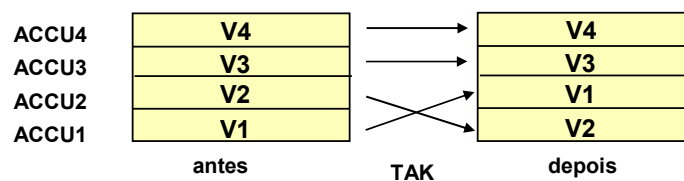
As funções de acumulador permitem otimizar o tempo de execução de programas de automação.

## A Instrução TAK (Troca ACCU1 com ACCU2)

### S7-300:



### S7-400:



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.3Conhecimento em Automação  
Training Center

### TAK

TAK (troca ACCU1 com ACCU2) troca o conteúdo do ACCU1 com o conteúdo do ACCU2. A instrução é executada sem levar em conta, e sem afetar, os bits de status. O conteúdo de ACCU3 e ACCU4 permanecem inalterados para CPUs com quatro acumuladores (para S7-400).

### Exemplo

Subtrair o valor menor do valor maior:

```

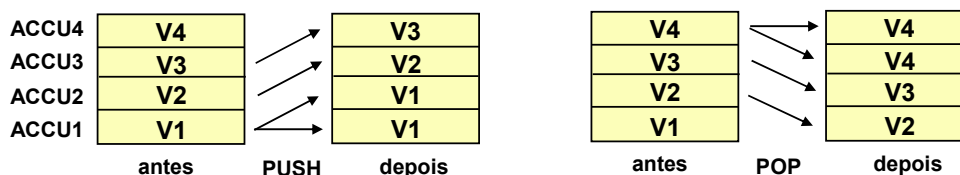
L  MW10    // Carrega o conteúdo do MW10 no ACCU1-L.
L  MW12    // Carrega o conteúdo do ACCU1-L no ACCU2-L.
           // Carrega o conteúdo do MW12 no ACCU1-L.
>I         // Cheque se ACCU2-L (MW10) é maior que ACCU1-L
           // (MW12).
JC NEXT    // Salta para o rótulo NEXT se ACCU2 (MW10) for
           // maior que ACCU1 (MW12).
TAK        // Troca o conteúdo do ACCU1 com ACCU2.
NEXT:-I    // Subtrai o conteúdo do ACCU1-L do conteúdo
           // do ACCU2-L.
T  MW14    // Transfira o resultado (= valor maior menos
           // valor menor) para MW14
  
```

## As Instruções PUSH e POP

### S7-300:



### S7-400:



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.4Conhecimento em Automação  
Training Center

### PUSH

A instrução o PUSH troca os conteúdos dos acumuladores, em cada caso, no próximo acumulador mais alto. PUSH normalmente é usado para duplicar o valor de ACCU1, sem perder os conteúdos originais de ACCU2 ou ACCU3 (só para S7-400).

- PUSH (S7-300): A instrução PUSH copia o conteúdo inteiro de ACCU1 para ACCU2. ACCU1 permanece inalterado.
- PUSH (S7-400): A instrução PUSH copia o conteúdo de ACCU3 para ACCU4, o conteúdo de ACCU2 para ACCU3 e o conteúdo de ACCU1 para ACCU2. ACCU1 permanece inalterado.

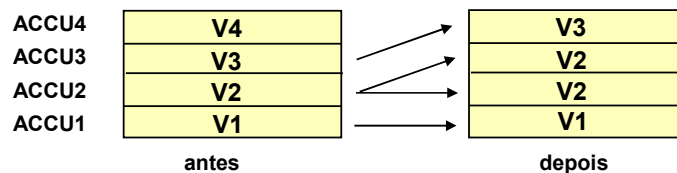
### POP

A instrução o POP traz os valores encontrados nos acumuladores 2 a 4 para os acumuladores subjacentes. Esta instrução normalmente é executada depois de instruções de transferência, quando o conteúdo do ACCU1 já não será mais necessário e o processamento continuará com os valores salvos nos acumuladores superiores.

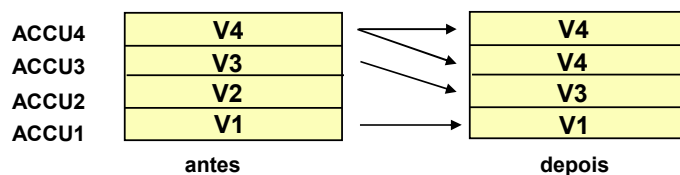
- POP (S7-300): A instrução POP copia o conteúdo inteiro de ACCU2 para ACCU1. ACCU2 permanece inalterado.
- POP (S7-400): A instrução POP copia o conteúdo de ACCU2 para ACCU1, o conteúdo de ACCU3 para ACCU2 e o conteúdo de ACCU4 para ACCU3. ACCU4 permanece inalterado.

## As Instruções ENT e LEAVE (só S7-400)

### ENT:



### LEAVE:



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.5



Conhecimento em Automação  
Training Center

### ENT

A instrução ENT (ENTER ACCU Stack) desloca os conteúdos dos acumuladores 2 e 3 respectivamente para os acumuladores 3 e 4. Os conteúdos de acumuladores 1 e 2 permanecem inalterados.

ENT junto com uma função LOAD imediatamente seguinte:

- ENT
- L ...

Tem como resultado, que durante o carregamento os conteúdos dos acumuladores 1 a 3 serão deslocados para cima (semelhante a PUSH) e o valor do ACCU1 permanece inalterado.

A instrução ENT é executada sem levar em conta e sem afetar os bits de status.

### LEAVE

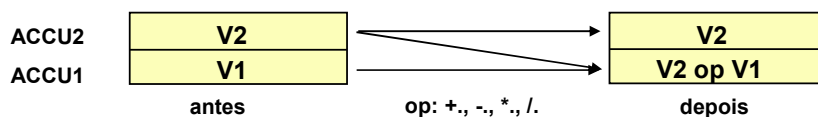
A instrução LEAVE desloca os conteúdos dos acumuladores 4 e 3 respectivamente para os acumuladores 3 e 2. Os conteúdos de acumuladores 4 e 1 permanecem inalterados.

As funções aritméticas utilizam do recurso LEAVE. Com LEAVE você pode também emular a mesma funcionalidade em outras operações de lógica digitais (por exemplo uma instrução de lógica de palavra).

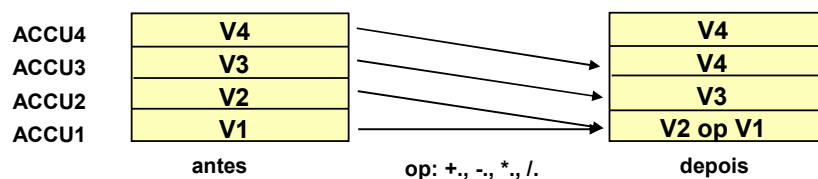
LEAVE programada depois de uma operação de lógica digital, vai transferir os conteúdos de acumuladores 3 e 4 para os acumuladores 2 e 3. O resultado da operação de lógica digital se mantém inalterado no acumulador 1.

## Instruções Aritméticas

### S7-300:



### S7-400:



## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.6



Conhecimento em Automação  
Training Center

### Instruções Aritméticas

As instruções aritméticas combinam dois valores digitais encontrados nos acumuladores 1 e 2, conforme os fundamentos das operações aritméticas. O resultado do cálculo fica no acumulador 1. Os bits de status CC0, CC1, OV e OS fornecem informações sobre o resultado final ou intermediário do cálculo.

### S7-300

Nas CPUs do S7-300, o conteúdo do ACCU2 permanece inalterado com a execução de uma função aritmética.

### S7-400

Com a CPUs S7-400, o conteúdo de ACCU2 é sobrescrito pelo conteúdo do ACCU3. O conteúdo de ACCU4 é transferido para ACCU3.

### Exemplo

O segmento de programa seguinte produz diferentes resultados, dependendo de qual tipo de CPU é executado, em uma S7-300 ou em uma S7-400 :

```
L 0 // carrega o inteiro 0 no ACCU1
L 5 // carrega o inteiro 5 no ACCU1, 0 no ACCU2
PUSH // desloca 5 (ACCU1) para ACCU2; (S7-400: ACCU2->ACCU3)
*I // multiplica ACCU1 por ACCU2; (S7-400: ACCU3->ACCU2)
*I // multiplica ACCU1 por ACCU2; (S7-400: ACCU3->ACCU2)
```

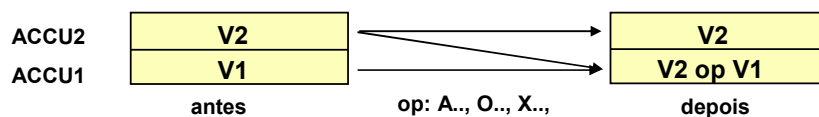
#### Resultado:

S7-300: ACCU1 = 125

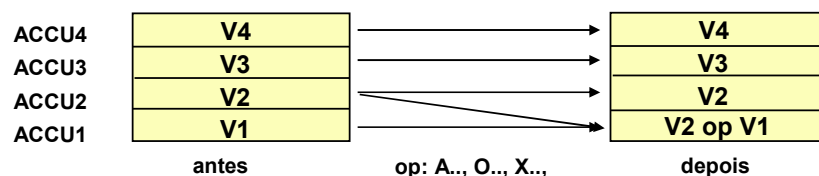
S7-400: ACCU1 = 0

## Instruções Lógicas de Palavras

### S7-300:



### S7-400:



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.7



Conhecimento em Automação  
Training Center

### Instruções Lógicas de Palavras

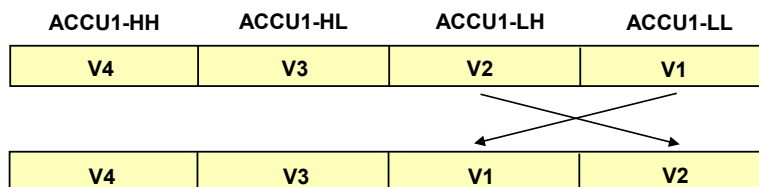
As instruções lógicas de palavras combinam bit a bit os valores do ACCU1 com uma constante ou com o conteúdo do ACCU2 e guarda o resultado no ACCU1.

O conteúdo dos acumuladores restantes (ACCU2 para S7-300, ou ACCU2, ACCU3 e ACCU4 para S7-400) permanecem inalterados. As operações lógicas podem ser executadas para formatos de palavras (Word) ou palavras duplas (Double Word).

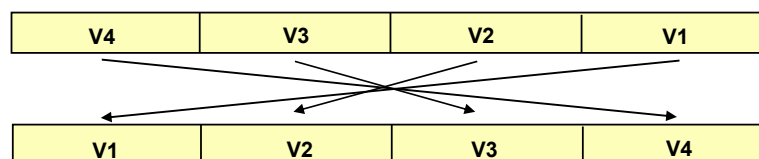
As instruções AND, OR ou Exclusive OR são disponíveis como instruções lógicas de palavras.

## Instruções de Troca no ACCU1

### CAW:



### CAD:



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.8



Conhecimento em Automação  
Training Center

### CAW

Com a instrução CAW, os bytes na palavra de dados direita do ACCU1 são trocados entre si, ou seja, o conteúdo do ACCU1-LH é transferido para o ACCU1-LL e vice versa.

Com esta instrução é possível converter um número de 16-bit (INT ou WORD) em representação SIMATIC para o formato de representação numérica INTEL (transferência de dados para PCs).

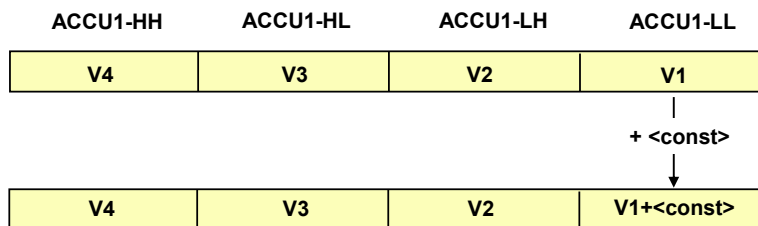
### CAD

Com a instrução CAD, os bytes no ACCU1 são trocados entre si, ou seja, o conteúdo do ACCU1-HH é transferido para ACCU1-LL e vice versa e o conteúdo do ACCU1-HL para ACCU1-LH e vice versa.

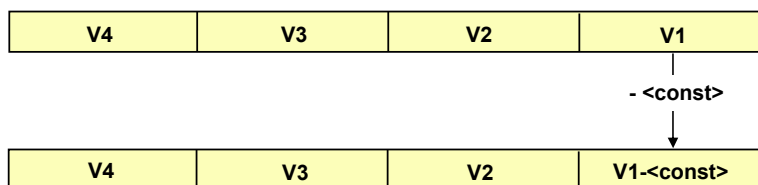
Com esta instrução é possível converter um número de formato 32-bit (DINT, DWORD ou DINT) em representação SIMATIC para o formato de representação numérica INTEL (transferência de dados para PCs).

## Instruções Incrementais no ACCU1

**INC <const>:**



**DEC <const>:**



**SIMATIC S7**

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.9



Conhecimento em Automação  
Training Center

### INC

A instrução INC <inteiro de 8 bits> adiciona um inteiro de 8 bits ao conteúdo do ACCU1-LL e salva o resultado no ACCU1-LL.

ACCU1-LH, ACCU1-H e ACCU2, ou ACCU3 e ACCU4 permanecem inalterados.

### DEC

A instrução DEC <inteiro de 8 bits> subtrai um inteiro de 8 bits do conteúdo do ACCU1-LL e salva o resultado no ACCU1-LL.

ACCU1-LH, ACCU1-H e ACCU2, ou ACCU3 e ACCU4 permanecem inalterados.

### Notas

As instruções INC e DEC são também conhecidas como "Instruções de Baixo Nível", isto é, no caso de um estouro (overflow), o processador não gera qualquer bit de estouro na palavra de status.

Em vez da instrução INC, você pode também, por exemplo, usar as seguintes instruções para soma de INT- ou DINT-:

+ <const> (soma uma <const> 16-bit ao ACCU1)

+ L#<const> (soma uma <const> 32-bit ao ACCU1)

Em vez da instrução DEC, você pode também, por exemplo, usar as seguintes instruções para subtração de INT- ou DINT-:

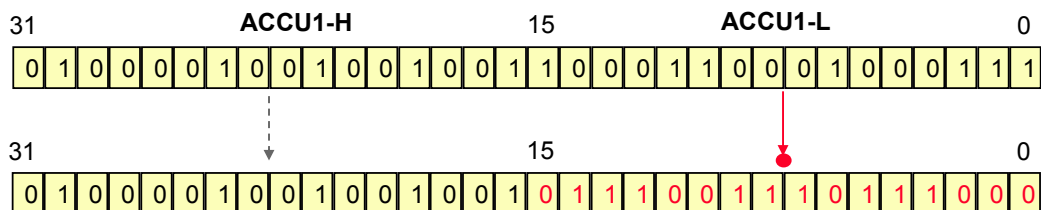
+ - <const> (subtrai uma <const> 16-bit do conteúdo do ACCU1)

+ L# - <const> (subtrai uma <const> 32-bit do conteúdo do ACCU1)

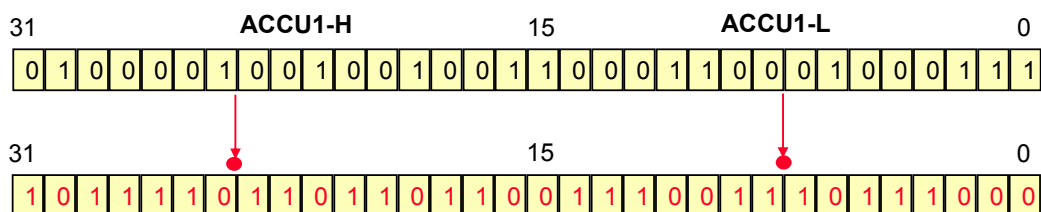


## Formando o Complemento de Um

### INVI (Complemento de um do ACCU1-L):



### INVD (Complemento de um do ACCU1):



#### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.10



Conhecimento em Automação  
Training Center

#### INVI

A instrução INVI nega bit a bit o conteúdo (bits 0 a 15) da palavra direita do acumulador 1. Ela troca zero's por um's e um's por zero's. O conteúdo da palavra esquerda (bits 16 to 31) permanece inalterado.

A instrução INVI não modifica qualquer bit de status.

#### INVD

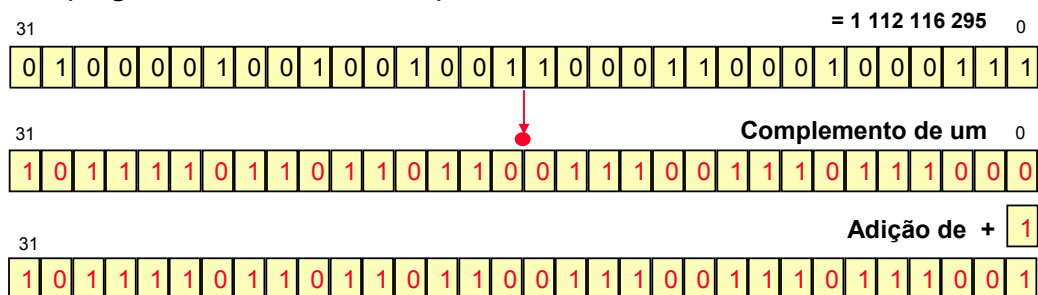
A instrução INVD nega bit a bit o conteúdo (bits 0 to 31) do acumulador 1. Ela troca zero's por um's e um's por zero's.

A instrução INVD não modifica qualquer bit de status.

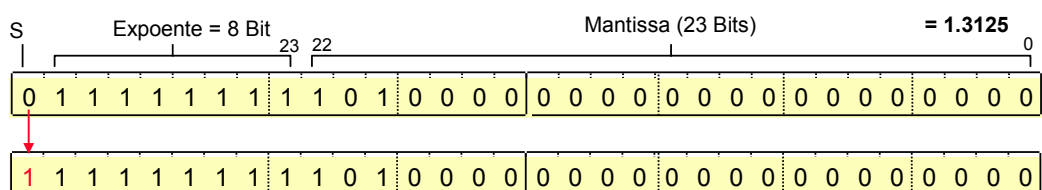
## Negação de Números (Complemento de Dois)

### NEGI (Negação de números INT)

#### NEGD (Negação de números DINT):



#### NEGR (Negação de número REAL):



SIMATIC S7

Siemens AG 1998. All rights reserved.

OUT

Date: 4/10/2007  
File: PRO2\_02P.11Conhecimento em Automação  
Training Center

#### NEGI

A instrução NEGI interpreta o valor (bits 0 a 15) encontrado na palavra direita do ACCU1 como um número INTEIRO e através da formação do complemento de dois reverte o sinal.

Esta instrução equivale a multiplicação por "-1". A palavra esquerda do ACCU1 (bits 16 to 31) permanece inalterada.

Os bits de status CC1, CC0, OS e OV são setados como uma função do resultado da operação.

#### NEGD

A instrução NEGD interpreta o valor encontrado no ACCU1 como um número duplo inteiro e o multiplica por "-1".

A formação do complemento de dois também pode ser feita através do "complemento de um" somado de "+1".

Os bits de status CC1, CC0, OS e OV são setados como uma função do resultado da operação.

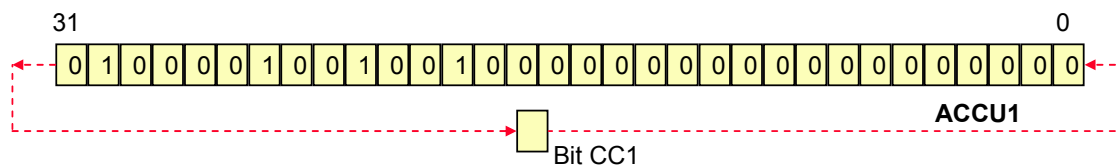
#### NEGR

A instrução NEGR interpreta o valor encontrado no ACCU1 como um número REAL (32 bit, IEEE-FP) e multiplica este número por "-1". A instrução inverte o estado do bit 31 no ACCU1 (sinal da mantissa).

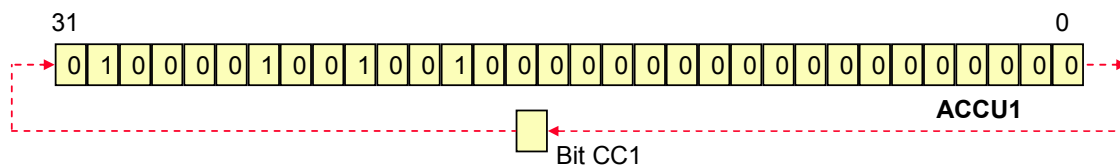
A instrução NEGR não modifica os bits de status.

## Instruções de Rotação em 32 Bits via Bit CC1

### RLDA (Rotação esquerda via status bit CC1):



### RRDA (Rotação direita via status bit CC1):



## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.12Conhecimento em Automação  
Training Center

### RLDA

A função RLDA desloca o conteúdo do ACCU1 em 1 casa para esquerda. A posição do bit 0 será preenchida com o valor do status bit CC1. O status bit CC1 ficará com o conteúdo do bit 31.

Os bits de status CC0 e OV são zerados ("0").

- Exemplo:

ACCU1: Y100 0100 1100 0100

CC1: X

RLDA

ACCU1: 1000 1001 1000 100X

CC1: Y

### RRDA

A função RRDA desloca o conteúdo do ACCU1 em 1 casa para direita. A posição do bit 31 será preenchida com o valor do status bit CC1. O bit de status CC1 ficará com o conteúdo do bit 0.

Os bits de status CC0 e OV são zerados ("0").

- Exemplo:

ACCU1: 0100 0100 1100 010Y

CC1: X

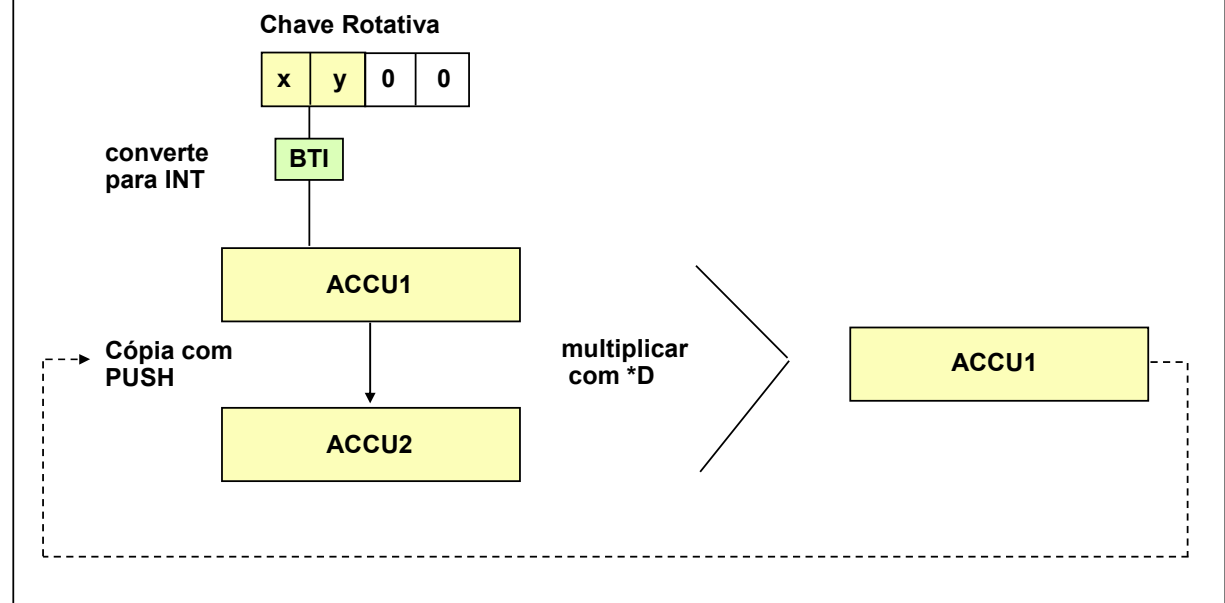
RRDA

ACCU1: X010 0010 0110 0010

CC1: Y

## Exercício 2.1: Cálculo com Expoentes

Exemplo : Formação da 6ª potência de um número inteiro através de sucessivas instruções PUSH e \*I



SIMATIC S7  
Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.13



Conhecimento em Automação  
Training Center

### Objetivo

Familiarizar-se com instruções de troca de acumuladores através do cálculo de potenciação de números inteiros.

### Enunciado

Gerar a FC21 com a seguinte funcionalidade:.

- Ler o byte esquerdo da chave rotativa e converter o valor BCD em valor Inteiro (BTI).
- Calcular a 6ª potência do valor lido.

### Procedimento

1. Copiar o conteúdo do ACCU1 no ACCU2 com a ajuda do comando PUSH.
2. Multiplicar o ACCU1 por ACCU2 (cálculo do quadrado).
3. Copiar o conteúdo do ACCU1 no ACCU2 com a ajuda do comando PUSH.
4. etc., etc., etc.

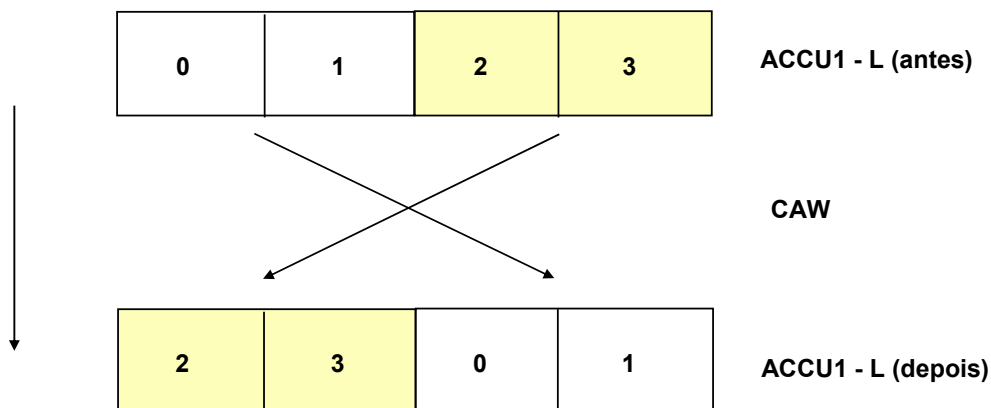
Cuidado: Como o quarto passo deve ser tratado para que a FC21 retorne o resultado correto tanto para um S7-300 como para um S7-400?

5. Apresente o resultado no display digital.
6. Chame a FC21 no OB1 e faça o download do programa para a CPU.
7. Teste o programa.

### Nota

Para que o valor lido não seja muito grande, somente a década da direita deve ser lida.

## Exercício 2.2 : Troca de Dados no ACCU1



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.14



Conhecimento em Automação  
Training Center

### Objetivo

Familiarizar-se com troca de bytes no ACCU1.

Aplicação: Converter um número em representação SIMATIC para uma representação de PC usando INTEL-CPU (80486, Pentium,...).

Esta conversão é muito útil quando se faz troca de dados entre um controlador SIMATIC e um PC.

### Enunciado

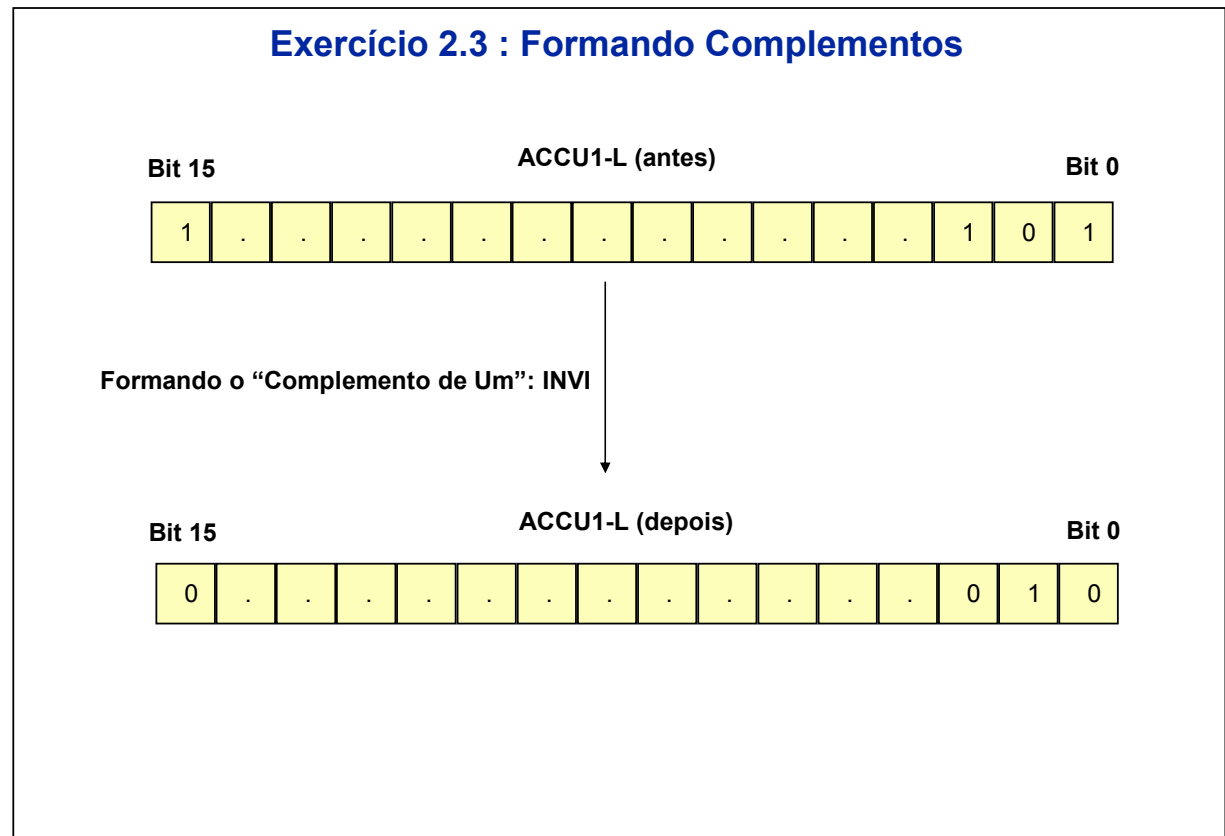
Gerar a FC22 com a seguinte funcionalidade:

- Carregue o valor da chave rotativa no ACCU1.
- Troque os 2 bytes do ACCU1-L com auxílio do comando CAW.
- Apresente o conteúdo do ACCU1 no display.

### Procedimento

1. Gerar a FC22.
2. Chamar a FC22 no OB1.
3. Fazer o download do programa para a CPU S7.
4. Testar o programa.

## Exercício 2.3 : Formando Complementos



**SIMATIC S7**  
Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_02P.15



Conhecimento em Automação  
Training Center

### Objetivo

Familiarizar-se com a instrução de formação de complemento no SIMATIC S7.

Aplicação: Converter um "negativo" em um "positivo "

Tais conversões são sempre empregadas quando se lê sinais ativos em "0", porém o programa de usuário trabalha com lógica positiva ou quando sinais invertidos (0-ativo) são enviados para os equipamentos de campo.

### Enunciado

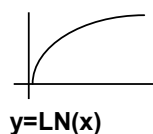
Gerar a FC 23 com a seguinte funcionalidade:

- Carregue o valor da palavra das chaves de teste de entrada no ACCU1.
- Forme o complemento de um.
- Apresente o resultado nos LED's do simulador.

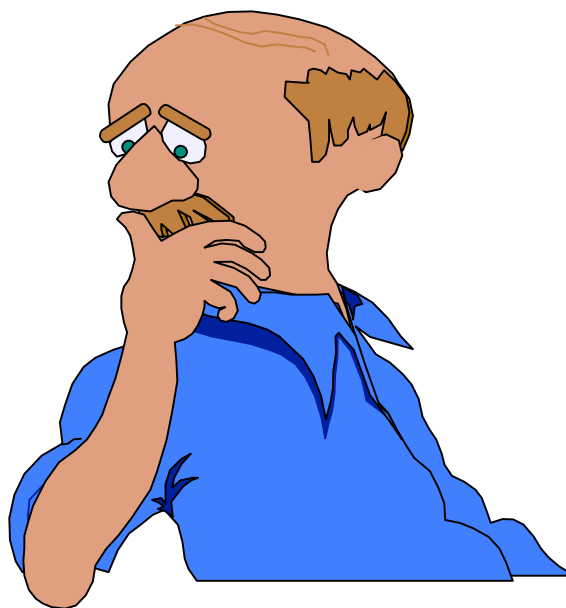
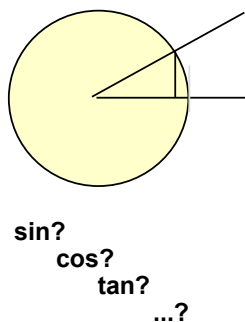
### Procedimento

1. Gerar a FC 23
2. Chamar a FC 23 no OB 1
3. Fazer download do programa para a CPU S7.
4. Testar o programa com auxílio das variáveis de status e opções de preset binário.

## Instruções com números REAIS



??



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.1



Conhecimento em Automação  
Training Center

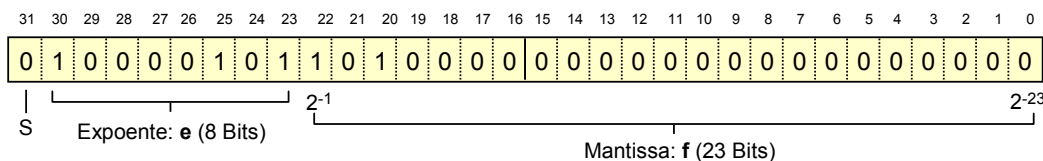
### Conteúdo

Pág.

Representação de números REAIS no SIMATIC S7 .....	2
Instruções Básicas com números REAIS .....	3
Funções Matemáticas Adicionais .....	4
Funções Trigonômicas e suas Funções Inversas .....	5
Outras Instruções com números REAIS .....	6
Exercício 3.1: Calculando Distância .....	7

## Representação de números REAIS no SIMATIC S7

- Formato da representação de um número REAL (IEEE FP formato binário 32 bits):



- Representação de um número REAL normalizado:

$$S \times (1.f) \times 2^{(e-127)}$$

S = Bit de sinal (0 corresponde a "+", 1 corresponde a "-")

f = 23 bits da Mantissa com MSB =  $2^{-1}$  e LSB =  $2^{-23}$

e = expoente binário inteiro ( $0 < e < 255$ )

- Exemplo:

S = 0

e = 1000 0101 = 133

f = 1010 0000... = 0.5 + 0.125



$$R = +1.625 \times 2^{(133-127)} = 1.625 \times 64 = 104.0$$

- Faixa de valores dos números REAIS normalizados:

- 3.402 823 x  $10^{+38}$  ... -1.175 494 x  $10^{-38}$ , 0, 1.175 494 x  $10^{-38}$  ... 3.402 823 x  $10^{+38}$

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.2



Conhecimento em Automação  
Training Center

### Número REAL

Os números REAIS (ponto flutuante) habilitam a implementação de complexos cálculos matemáticos para controle de processos e controle de processos em malha fechada.

Uma variável tipo dado REAL consiste internamente de três componentes: o sinal, o expoente 8 bits em base 2 e a mantissa 23 bits.

O sinal pode assumir os valores "0" (positivo) ou "1" (negativo). O expoente é incrementado por uma constante (Bias, +127) e armazenado, então este terá uma faixa de valores de 0 a 255.

A mantissa representa a parte fracionária. A parte inteira da mantissa não é armazenada, uma vez que esta será sempre 1 (para ponto flutuante normalizado) ou 0 (para ponto flutuante não normalizado).

### Limites de faixa

Designação	Valor e	Mantissa f	Valor	CC1	CC0	OV	OS
No.não pt.flut.	255	<>0	[qNaN]	1	1	1	1
Estouro	255	0	$>(2-2^{-23}) 2^{127}$ $<(-2+2^{-23}) 2^{127}$	1 0	0 1	1 1	1 1
No. normalizado	1.. 254	qualquer	$(1.f) 2^{e-127}$ $(-1.f) 2^{e-127}$	1 0	0 1	0 0	- -
No.não normaliz.	0	<>0	$(0.f) 2^{-126}$ $(-0.f) 2^{-126}$	0 0	0 0	1 1	1 1
Zero	0	0	+0	0	0	0	-

### Nota

As CPUs calculam com inteira exatidão os números em ponto flutuante. O display na PG pode desviar-se da exata representação, devido ao erro de arredondamento para cima na conversão. Números REAIS são arredondados para cima a partir da sexta casa decimal.



## Instruções Básicas com números REAIS

- **Adição em REAL:**

L	MD10	// Carrega o primeiro número REAL
L	MD20	// Carrega o segundo número REAL
<b>+R</b>		// Soma os números REAIS (MD10 + MD20)
T	MD30	// Transfere o resultado para o MD30

- **Subtração em REAL:**

L	MD10	// Carrega o primeiro número REAL
L	MD20	// Carrega o segundo número REAL
<b>-R</b>		// Subtrai os números REAIS (MD10 - MD20)
T	MD30	// Transfere o resultado para o MD30

- **Multiplicação em REAL:**

L	MD10	// Carrega o primeiro número REAL
L	MD20	// Carrega o segundo número REAL
<b>*R</b>		// Multiplica os números REAIS (MD10 * MD20)
T	MD30	// Transfere o resultado para o MD30

- **Divisão em REAL:**

L	MD10	// Carrega o primeiro número REAL
L	MD20	// Carrega o segundo número REAL
<b>/R</b>		// Divide os números REAIS (MD10 / MD20)
T	MD30	// Transfere o resultado para o MD30

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.3Conhecimento em Automação  
Training Center

### Vista Geral

As funções +R, -R, \*R, /R interpretam os valores encontrados no ACCU1 e ACCU2 como números tipo dados REAIS. Eles executam a operação lógica programada (+R, -R, \*R and /R) e salvam o resultado no ACCU1.

Após os cálculos serem executados, os bits de status CC0 e CC1 indicam, se o resultado é negativo (CC1=0, CC0=1), zero (CC1=0; CC0=0) ou positivo (CC1=1, CC0=0).

Os bits de status OV e OS sinalizam se as operações não excederam a faixa de números permitidos.

### Números REAIS não autorizados

Com um cálculo não autorizado, isto é, quando um dos dois valores inseridos é um número REAL inválido, então o resultado no ACCU1 é também um número REAL inválido.

Números REAIS inválidos são também armazenados como um resultado no ACCU1, se você tentar processar valores não autorizados com as seguintes instruções:

Adição: Soma de + infinito e - infinito.

Subtração: Subtração de + infinito e + infinito  
ou - infinito e - infinito.

Multiplicação: Multiplicação de 0 por infinito.

Divisão: Divisão de infinito por infinito ou 0 por 0.

O resultado da divisão de números REAIS válidos por 0 é, dependendo do sinal do número, + infinito ou - infinito.

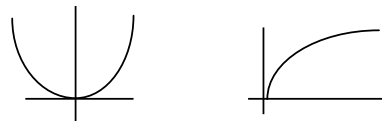
### Nota

O número hexadecimal D#16#FFFF FFFF representa, por exemplo, um número REAL inválido.

## Funções Matemáticas Adicionais

### Funções Matemáticas:

SQR	Forma o quadrado de um número
SQRT	Calcula a raiz quadrada
EXP	Função exponencial na base "e"
LN	Logarítmo natural (e=2.718282)

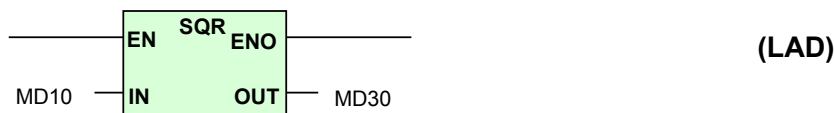


### Exemplo:

```

L      MD10      // Carrega um número REAL
SQR    MD30      // Calcula o quadrado           (STL)
T      MD30      // Transfere o resultado para MD30

```



## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.4



Conhecimento em Automação  
Training Center

### Vista Geral

As funções matemáticas pegam o número no ACCU1 como o valor de entrada da função a ser executada e armazena o resultado no ACCU1.

Funções matemáticas somente mudam o conteúdo do ACCU1. O conteúdo do ACCU2, ou ACCU3 e ACCU4 para S7-400, permanecem inalterados.

Dependendo do resultado da função, a função matemática seta os bits de status CC0, CC1, OV e OS.

Se existe um número REAL inválido no ACCU1 antes da função ser executada, então a função matemática retorna um número REAL inválido e seta os bits de status correspondentemente.

### SQR

A função SQR eleva ao quadrado o conteúdo do ACCU1.

### SQRT

A função SQRT calcula a raiz quadrada do valor no ACCU1. Se existe um valor menor do que zero no ACCU1, SQRT seta os bits de status CC0, CC1, OV e OS para "1" e retorna um número REAL inválido.

Se -0 (menos zero) está no ACCU1, -0 também é retornado.

### EXP

A função EXP calcula a potência na base "e" (e=2.71828) e o valor ( $e^{\text{ACCU1}}$ ) encontrado no ACCU1.

### LN

A função LN calcula o logarítmo natural para base "e" do número encontrado no ACCU1. Se existe um valor menor que ou igual a zero no ACCU1, LN seta os bits de status CC0, CC1, OV e OS para "1" e retorna um número REAL inválido.

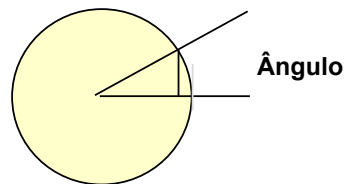
O logarítmo natural é a função inversa da função exponencial:

Se:  $y = e^x$   
então:  $x = \ln y$

## Funções Trigonométricas e suas Funções Inversas

- **Funções Trigonométricas:**

SIN	Seno
COS	Coseno
TAN	Tangente



- **Funções Arco:**

ASIN	Arco seno
ACOS	Arco coseno
ATAN	Arco tangente

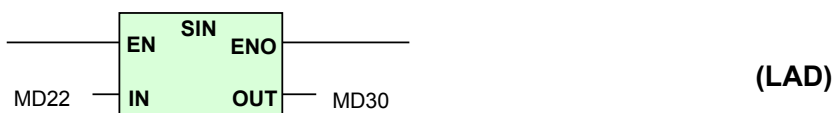
- **Exemplo:**

```

L      MD10    // Carrega um número REAL
SIN      // Calcula o seno
T      MD30    // Transfere o resultado para MD30

```

**(STL)**



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.5



Conhecimento em Automação  
Training Center

### Funções

#### Trigonométricas

As funções trigonométricas esperam por um ângulo em radianos medido como número REAL no ACCU1. Para o ângulo inserido ( $0^0 \dots 360^0$ ), você deve, se necessário, realizar uma conversão para graus medidos ( $0 \dots 2\pi$ , com  $\pi=3.141593$ ).

Durante a execução da função, para valores menores que 0 ou maiores que  $2\pi$ , um múltiplo de  $2\pi$  é automaticamente somado ou subtraído até que o valor se encontre entre 0 e  $2\pi$  (módulo automático de cálculo  $2\pi$ ).

#### Funções Arco

As funções arco são o inverso de suas respectivas funções trigonométricas. Elas esperam um número REAL em uma faixa específica de valores no ACCU1 e retornam um ângulo medido em radianos:

Função	Faixa permitida definida	Faixa de valores
ASIN	-1 a +1	- $\pi/2$ a + $\pi/2$
ACOS	-1 a +1	0 a $\pi$
ATAN	faixa inserida	- $\pi/2$ a + $\pi/2$

Com uma sobrefaixa da faixa permitida definida, as funções arco retornam um número REAL inválido e setam os bits de status CC0, CC1, OV e OS para "1".

## Outras Instruções com números REAIS

- **Instruções de conversão de REAL para DINT:**

RND+	arredonda para o próximo número DINT acima
RND-	arredonda para o próximo número DINT abaixo
RND	arredonda para o inteiro mais próximo
TRUNC	trunca o número, mantendo somente a parte inteira

- **Instruções de conversão de DINT para REAL:**

DTR	converte com arredondamento
-----	-----------------------------

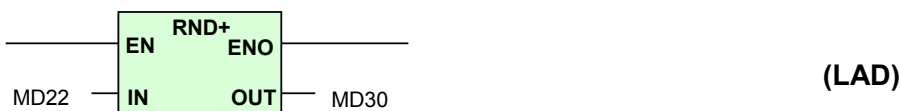
- **Outras instruções de REAL para REAL:**

ABS	retorna o valor absoluto ou módulo
NEGR	nega um número REAL

- **Exemplo:**

```

L      MD10      // Carrega um número REAL
RND+   // Converte para o próximo número DINT acima  (STL)
T      MD30      // Transfere o resultado para MD30
  
```



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.6Conhecimento em Automação  
Training Center

### Vista Geral

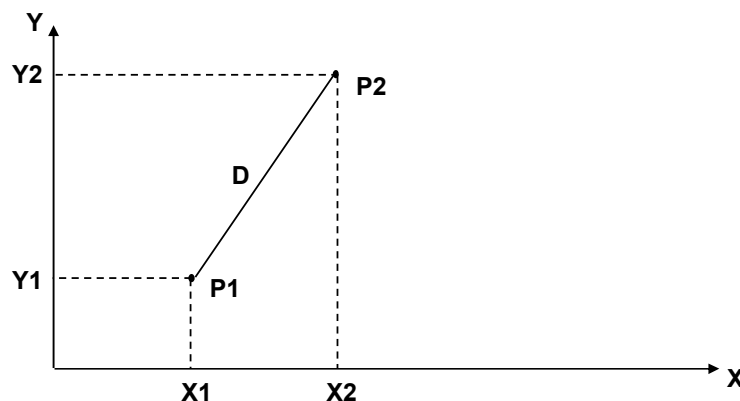
As funções de conversão convertem os tipos de dados dos valores encontrados no ACCU1 em outro tipo de dado e armazena o resultado no ACCU1. O conteúdo dos outros acumuladores permanecem inalterados.

Se, em uma das instruções (RND+, RND-, RND ou TRUNC), o valor encontrado no ACCU1 é maior ou menor do que a faixa de formatos permissíveis de DINT ou este não corresponde ao número em formato REAL, a instrução seta os bits de status OV e OS para "1". Uma conversão então não tem lugar.

<b>RND+</b>	A instrução RND+ converte o conteúdo do ACCU1 como número REAL em um inteiro (DINT), o qual é maior ou igual ao número a ser convertido.
<b>RND-</b>	A instrução RND- converte o conteúdo do ACCU1 como número REAL em um inteiro (DINT), o qual é menor ou igual ao número a ser convertido.
<b>RND</b>	A instrução RND converte o conteúdo do ACCU1 como número REAL no próximo inteiro possível (DINT). Se o resultado estiver exatamente entre um número par e um número ímpar, o número par é retornado.
<b>TRUNC</b>	A instrução TRUNC retorna o componente inteiro do número a ser convertido; a parte fracionária é jogada fora.
<b>DTR</b>	A instrução DTR converte um número de formato DINT para o formato de número REAL. Uma vez que um número em formato DINT é mais exato do que um número em formato REAL, é possível que durante a conversão um arredondamento tenha lugar para o próximo número representável.
<b>ABS</b>	A instrução ABS forma o valor absoluto do número REAL encontrado no ACCU1, isto é, o sinal (bit 31) é fixado em "0" (par para um número REAL inválido).
<b>NEGR</b>	A instrução NEGR nega o número REAL no ACCU1, isto é, o sinal (bit 31) é invertido (par para um número REAL inválido). As instruções DTR, ABS e NEGR não afetam os bits de status.

### Exercício 3.1: Calculando Distância

**Exemplo: Calculando a distância D entre dois pontos em um sistema de coordenadas retangulares**



Função: FC 31 com  $D = \sqrt{(X2 - X1)^2 + (Y2 - Y1)^2}$

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_3P.7



Conhecimento em Automação  
Training Center

#### Objetivo

A aplicação de funções matemáticas para cálculo de distância entre dois pontos.

#### Tarefa

Criar um FC31 com a seguinte funcionalidade:

- FC31 espera as coordenadas (X1, Y1) ou (X2, Y2) de dois pontos P1 e P2 nos parâmetros de entrada.
- FC31 retorna a distância entre os dois pontos no parâmetro de saída RET\_VAL.
- FC31 deverá ser instalável no sistema S7-300 bem como no sistema S7-400. Este não deverá utilizar endereços globais de CPU para qualquer possível salvamento dos resultados imediatos.

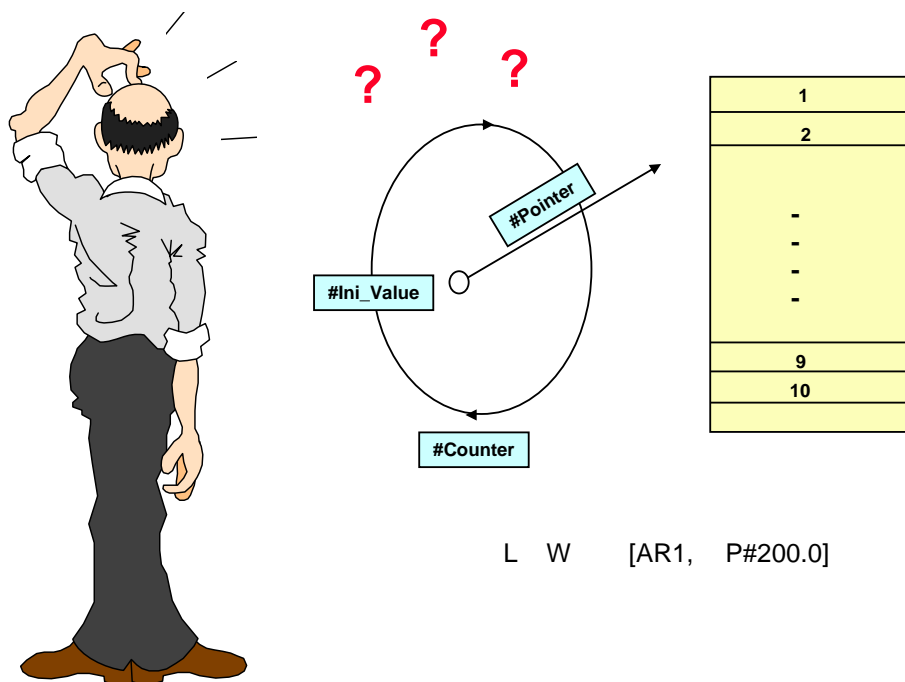
#### O que fazer

1. Criar um FC31 com a seguinte funcionalidade.
2. Chamar FC31 no OB1 e conectar os parâmetros de entradas e saídas como a seguir:  
X1 = MD0, Y1 = MD4  
X2 = MD8, Y2 = MD12  
RET\_VAL = MD16
3. Transferir o programa para a CPU S7.
4. Testar FC31 com ajuda da "Monitor/Modify Variable".

#### Additional Task

Criar uma versão run-time otimizada do FC31 para S7-400, que possa operar sem o uso de variáveis temporárias.

## Endereçamento Indireto e Instruções com Registrador de Endereços



SIMATIC S7

Siemens AG 1998. All rights reserved.

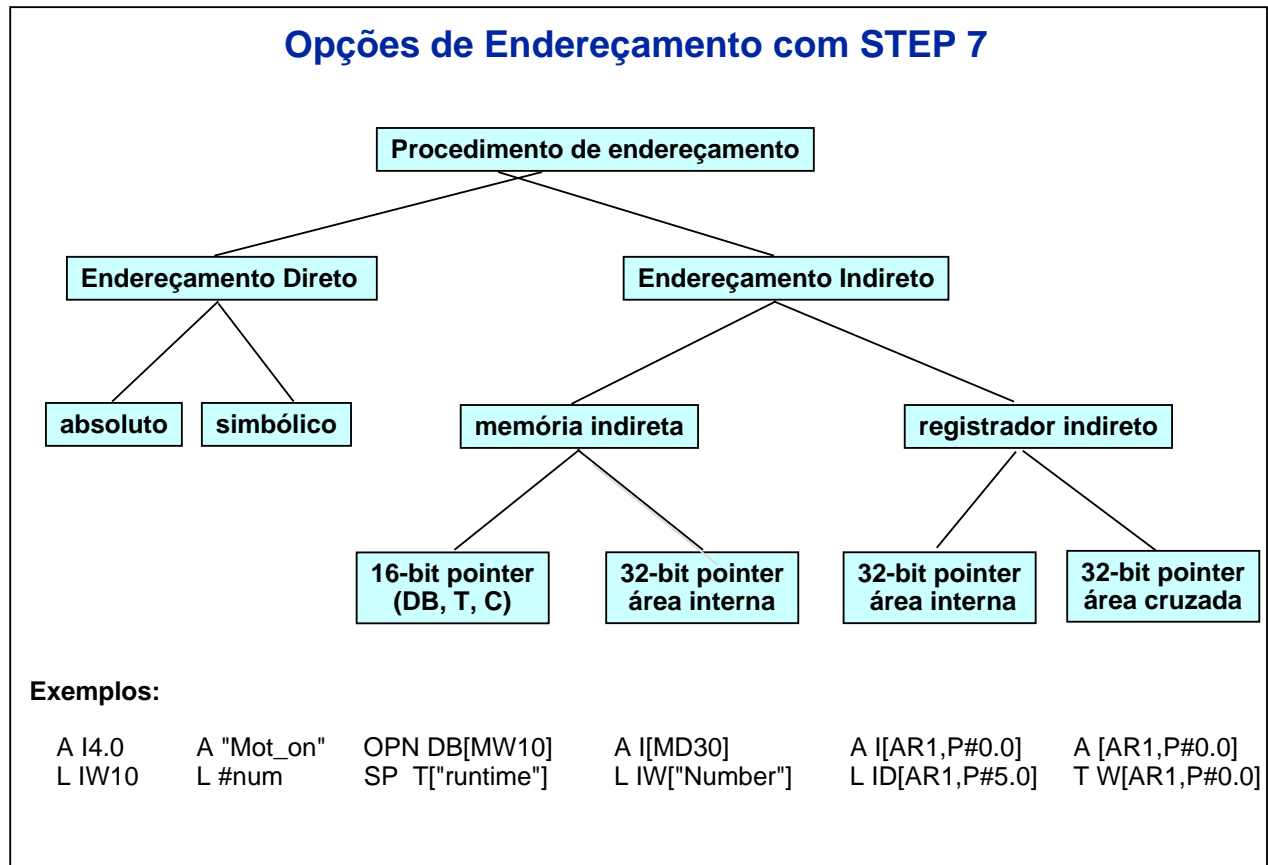
Date: 4/10/2007  
File: PRO2\_04P.1Conhecimento em Automação  
Training Center

### Conteúdo

Pág.

Opções de endereçamento com STEP7 .....	2
Endereçamento Direto de variáveis .....	3
Endereçamento Direto de variáveis em DBs .....	4
Avaliação de informação de DB no programa .....	5
Memória de Endereçamento Indireto .....	6
Estrutura de ponteiros com Endereçamento Indireto de Memória .....	7
Características do Endereçamento Indireto de Memória .....	8
Exemplo de Endereçamento Indireto .....	9
Exercício 4.1: Programação de Loop com Endereçamento Indireto .....	10
Endereçamento Indireto de Registrador de Área Interna .....	11
Endereçamento Indireto de Registrador de Área Cruzada .....	12
Instruções para carregamento dos Registradores de Endereço .....	13
Outras instruções com Registradores de Endereço .....	14
Características do Registrador de Endereçamento Indireto .....	15
Exercício 4.2: Loop com Registrador de Endereçamento Indireto .....	16
Tipos de ponteiros do STEP7 .....	17
Estrutura e atributos de um dado tipo "POINTER" .....	18
Configuração de um dado tipo "ANY" .....	19
Atributos dos parâmetros de dados tipo "ANY" .....	20
Atributos indiretos de parâmetros do tipo "ANY" .....	21
Avaliando um ponteiro tipo "ANY" .....	22
Exercício 4.3: Função para calcular valor da soma e média .....	23

## Opções de Endereçamento com STEP 7



### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.2



Conhecimento em Automação  
Training Center

### Endereçamento Direto

Com endereçamento direto, o local da memória é codificado na instrução, ou seja, o endereço especifica o valor do endereço com o qual a instrução deve processar.

### Endereçamento Simbólico

Num programa, endereços podem ser endereçados absolutamente (p.ex. I1.0) ou simbolicamente (p.ex. "start signal"). Os endereços simbólicos usam nomes ao invés de endereços absolutos. Um programa é facilmente interpretado quando são usados nomes com significado. Com endereçamento simbólico é feita uma diferenciação entre símbolos locais (tabela de declarações de um bloco) e globais (tabela de símbolos).

### Endereçamento Indireto

Com endereçamento indireto pode-se endereçar endereços que somente são definidos durante a execução do programa. Com endereçamento indireto, partes do programa podem ser executadas repetidas vezes (Loop) e cada varredura com um endereço diferente.

Com endereçamento indireto é feita diferenciação entre:

- endereçamento indireto de memória: Um ponteiro do endereço endereçado será encontrado em uma célula de memória da memória do usuário (p.ex. MD30).

Com endereçamento indireto de memória, as variáveis, na memória na qual o ponteiro de endereçamento de endereço está guardada, também pode ser atribuído um nome simbólico.

- endereçamento indireto de registro: O ponteiro do endereço endereçado é carregado em um dos dois registradores de endereços do S7 (AR1 e AR2) antes de serem acessados.

### Precaução

**Como no endereçamento indireto os endereços somente são definidos durante a execução do programa, deve-se tomar o cuidado de não sobrescrever áreas de memória que possam causar reações inesperadas do PLC.**

## Endereçamento Direto de Variáveis

Operando	Endereço do Operando (p. ex.)	Dimensões de Acesso Adicionais	Designação
I	37.4	Byte, word, double word	Entradas (Inputs)
Q	27.7	Byte, word, double word	Saídas (Outputs)
PIB	655	Byte, word, double word	Periferia de Entrada (Peripheral inputs)
PQB	653	Byte, word, double word	Periferia de Saída (Peripheral outputs)
M	55.0	Byte, word, double word	Bits de Memória (Bit memories)
T	114	--	Temporizadores (Timers)
C	13	--	Contadores (Counters)
DBX	2001.6	Byte (DBB), word (DBW), double word (DBD)	Dado endereçado via Registrador de DB
DIX	406.1	Byte (DIB), word (DIW), double word (DID)	Dado endereçado via Registrador de DI
L	88.5	Byte (LB), word (LW), double word (LD)	Pilha de Dados Local (Local data stack)

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.3



Conhecimento em Automação  
Training Center

### Endereçamento Direto de Variáveis

O endereçamento direto permite endereçar variáveis simples (elementares), variáveis com comprimento máximo de 4 bytes. Variáveis simples consistem de:

- um operando (p.ex.: "IB" para input byte)
- um endereço exato (localização de memória)(byte ou bit) da área de memória que é determinada pelo identificador de endereço.

Endereços ou variáveis simples também podem ser endereçadas de modo global por nomes simbólicos (tabela de símbolos).

### Periféricos

Para endereçamento periférico, diferente do S5, agora é necessário fazer uma distinção entre inputs (L PIW) e outputs (T PQW).

### Dados locais

Com STEP 7 é possível também ter acesso aos dados da pilha de endereços locais de cada bloco, por exemplo:

- A L 12.6 (lógica "E" do bit local 12.6)
- L LW 12 (Carrega a palavra de dados local 12 no ACCU1)

### DBX/DIX

Você pode também acessar variáveis simples contidas em DB's:

- A DBX 12.6 (lógica "E" do bit 12.6 do DB que estiver aberto).
- L DB5.DBW10 (Carrega o DBW10 do DB5 no ACCU1)

### Variáveis Complexas

Você pode acessar variáveis locais, que possua um tipo de dado complexo, como estruturas (struct) ou vetores (array), simbolicamente.

Acesso absoluto só é possível com componentes de variáveis complexas, das quais suas partes são dados tipo elementares.



## Endereçamento Direto de Variáveis em DBs

Abrir o DB	Carregar e Transferir no DB
OPN DB 19 OPN "Values"	L DBB 1      Carrega o byte de dado 1 do DB L DBW 2      Carrega palavra de dado 2 (byte 2/3) L 5          Carrega o número 5 T DBW 4      Transfere para a palavra 4 do DB L 'A'        Carrega o caracter A em ASCII L DIB28      Carrega o byte de dado 28 do DI ==I          Compara
OPN DI 20	A DBX 0.0      Lê conteúdo do bit 0, byte 0
<hr/>	
Instrução combinada (contém OPN DB..)	L DB19.DBW4      Carrega palavra 4 do DB 19  L "Values".Number_1      Acesso simbólico da variável Number_1. DB19 tem o nome simbólico "Values"  A DB10.DBX4.7      Lê conteúdo do bit 7, byte 4 do DB 10

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.4Conhecimento em Automação  
Training Center

### Visão Geral

A CPU dispõe de dois registradores de DB para processamento de dados. O endereço dos DBs correntemente em uso estão guardados nestes registradores. Antes de você acessar dados de um DB, você deve primeiro abri-lo via um destes dois registradores.

Abrir um DB pode ser feito explicitamente pelas seguintes instruções:

- Opn DBx ou OPN DIx

ou implicitamente com auxílio de endereçamento combinado com operando:

- L DBx.DBWy      (L DIx.DIWy não é possível!)

Neste caso o número do DB também é carregado no registrador de DB.

### Endereçamento

DBs são organizados byte a byte no STEP7. O acesso direto aos endereços pode ser em BIT, BYTE, WORD ou DWORD (como I/Q/M).

### Acesso Simbólico

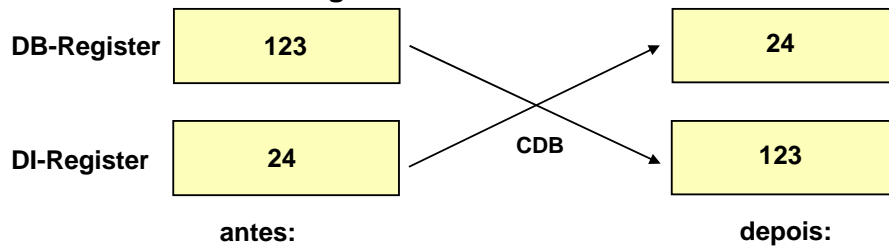
Para acesso simbólico deve-se entrar com o nome do símbolo do DB na lista de símbolos. Depois cria-se pelo editor de DB os elementos e seus respectivos símbolos.

Agora um completo acesso simbólico aos elementos de dados é possível com a instrução L "Values".Number\_1. Com isto, DB19 é aberto ("Values" é o nome simbólico do DB 19) e DBW 2 é carregado (Number\_1 é o nome simbólico do DBW2).

## Avaliação de Informação de DB no Programa

### ❑ Instruções com DB Registers:

- **CDB: Troca de DB Registers**



- **Carregar o DB Register no ACCU1**

- L DBNO (Carregue o numero do DB aberto no ACCU1)
- L DINO (Carregue o numero do DI aberto no ACCU1)

- **Carregar o comprimento dos DBs**

- L DBLG (carregue o comprimento/bytes do DB aberto no ACCU1)
- L DILG (carregue o comprimento/bytes do DI aberto no ACCU1)

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.5



Conhecimento em Automação  
Training Center

**Registradores DB, DI** Estes registradores contém o número dos DBs atualmente abertos. Dois Blocos de Dados podem ser abertos simultaneamente.

STL usa o primeiro registrador de DB preferencialmente para acessar o DB global e o segundo registrador de DB preferivelmente para acessar o DB instance. Estes registradores são também chamados de DB register ou DI register por esta razão.

A CPU trata os dois registradores igualmente. Todo DB pode ser aberto por um destes dois registradores (até mesmo via ambos simultaneamente).

**CDB** CDB (troca dos DB registers) troca o conteúdo dos registradores DB e DI. O conteúdo do registrador de DB é transferido para o registrador de DI e vice versa. Esta instrução não influencia o conteúdo do ACCU1 e nem os bits de status.

**L DBLG, L DILG:** Estas instruções lêem o comprimento em byte dos DBs abertos. Com a ajuda destas instruções o programa de usuário pode saber antes do DB ser acessado se ele possui o comprimento necessário.

**L DBNO, L DINO:** Estas instruções retornam o número dos DBs atualmente abertos.

## Memória de Endereçamento Indireto

### ❑ Ponteiro de 16-bit em formato Word (Endereçamento de DBs, T, C)

L 11

T MW 60

OPN DB[MW 60]



OPN DB 11

### ❑ Ponteiro de 32-bit em formato Double Word (Endereçamento de I, Q, M, ...)

L P#24.0

T MD 50

L I

W

[MD50]



L IW 24

Área

Dimensão  
do Acesso

Endereço

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.6Conhecimento em Automação  
Training Center

### Visão Geral

Com endereçamento indireto de memória, o endereço da variável a qual será acessada encontra-se em um endereço (localização de memória).

Programas que usam endereçamento indireto de memória contêm:

- uma instrução (p.ex.: OPN, A, L, etc.)
- um identificador de endereço (DB, C, T, I, QW, MD, etc.)
- e uma [variável], a qual tem que estar entre colchetes.

Esta variável contém o endereço (ponteiro) do operando a qual a instrução irá acessar.

Dependendo do identificador de endereço usado, a instrução interpretará o dado armazenado [variável] especificada, ou como um ponteiro de dimensão word ou double word.

### Instruções com Ponteiros 16-bit

Para endereçar temporizadores, contadores ou blocos (DB, FC, FB) use um ponteiro de 16 bits.

Todas as instruções de temporizadores ou contadores podem ser endereçadas usando endereçamento indireto. Para endereçar temporizadores, contadores ou blocos use identificadores de área dos formatos T, C, DB, DI, FB, FC. O endereço (localização de memória) do operando endereçado é armazenado em uma palavra.

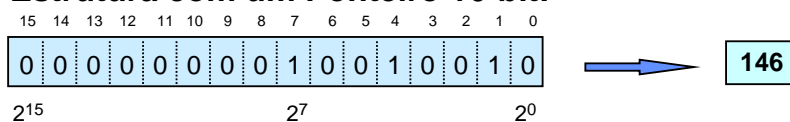
Um DB pode ser aberto via registrador de DB bem como um DI. Se um zero é encontrado no ponteiro quando você abre um bloco de dados (DB, DI) indiretamente, então o registrador DB/DI é carregado com o valor "0". Um erro não é gatilhado quando você carrega com "0".

A chamada dos blocos lógicos podem ser endereçados indiretamente com a ajuda das instruções UC ou CC (não CALL). Os blocos, contudo, não podem conter quaisquer parâmetros ou variáveis estáticas.

Este ponteiro em formato palavra (word) é interpretado como número inteiro (0 ... 65 535). Isto se refere ao número de um temporizador (T), um contador (C), um bloco de dados (DB, DI) ou um bloco lógico (FC, FB).

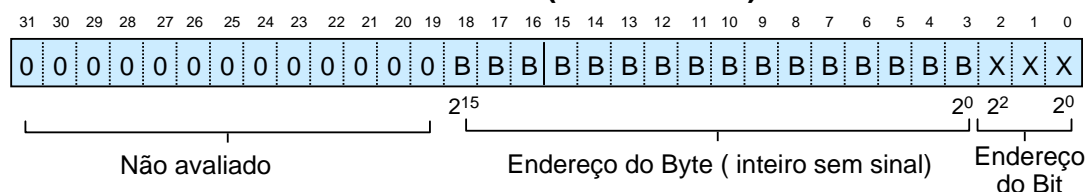
## Estrutura de Ponteiros com Endereçamento Indireto de Memória

### ❑ Estrutura com um Ponteiro 16-bit:



Interpretação como inteiro sem sinal entre 0 ... 65 535

### ❑ Estrutura com um Ponteiro 32-bit (área-interna):



### ❑ Carregamento de Constantes Ponteiro 32-bit (área-interna):

L P#25.3 (P = "Pointer", Endereço do Byte= 25, Endereço do Bit= 3)

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.7



Conhecimento em Automação  
Training Center

### Instrução com Ponteiros 32-bit

Os seguintes endereços podem ser acessados com auxílio de endereçamento indireto de memória via ponteiros 32-bit:

- Bits que são endereçados por operações de lógica binária. I, Q, M, L, DIX ou DBX podem ser usados como identificadores de endereço.
- Bytes, words e double words que podem ser endereçados por instruções Load ou Transfer IB, IW, ID, DBB, DBW, DBD, DIB, DIW, DID, PIB, PIW, PID, etc., podem ser usados como identificadores de endereço.

O endereço do operando endereçado é interpretado como ponteiro 32-bit. Nesta double word, os bits menos significativos (bit 0 a 2) são interpretados como endereço do bit, os próximos 16 bits (bit 3 a 18) são interpretados como endereço do byte. Bits 19 a 31 não são avaliados pela memória de endereçamento indireto.

### Nota

Se você quiser acessar um endereço por meio do endereçamento indireto de memória com instruções Load ou Transfer, você precisa se certificar de que o endereço de bit do ponteiro usado é "0".

Se não for este o caso, a CPU identificará um erro durante a execução.

### Carregando Constantes Ponteiro 32-bit

Uma constante ponteiro 32-bit pode ser carregada no ACCU1 com auxílio da seguinte sintaxe:

L P#<Endereço do Byte>.<Endereço do Bit>

### Localização dos Ponteiros

Ponteiros de 16-bit e 32-bit para endereçamento indireto de memória devem ser armazenadas em uma das seguintes opções:

- M - Memória M
- L - Dado local
- D - Bloco de Dados (DB ou DI)

## Características do Endereçamento Indireto de Memória

### □ Áreas de endereço para guardar Ponteiros de 16-bit e 32-bit:

- Memórias M (endereçamento absoluto ou simbólico, p.ex.: *OPN DB[MW30]*, *OPN DI["Motor\_1"]*, etc.  
*A I[MD30]*, *T QD["Speed\_1]*, etc.)
- Pilha de Dado Local (endereçamento absoluto ou simbólico, p.ex.: *OPN DB[LW10]*, *OPN DI[#DB\_NO]*, etc.  
*A I[LD10]*, *T QD[#Pointer]*, etc.)
- DBs globais (só endereçamento absoluto, DB deve ser aberto antes do uso, p.ex.: *OPN DB[DBW0]* (sobrescreve registrador DB !!!), *OPN DI[DBW22]*, etc.  
*A I[DBD10]*, *T QD[DBD22]*, etc.)
- DBs Instance (só endereçamento absoluto, DI deve ser aberto antes do uso, p.ex.: *OPN DB[DIW20]*, *OPN DI[DIW0]* (sobrescreve registrador DI !!!), etc.  
*A I[DID10]*, *T QD[DID22]*, etc.)

### □ Características na Passagem de Ponteiros para FBs e FCs

- Ponteiros passados por parâmetros não podem ser usados diretamente para endereçamento indireto de memória
- Ponteiros passados para endereçamento indireto de memória devem ser copiados em variáveis temporárias antes do acesso

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.8



Conhecimento em Automação  
Training Center

### Áreas de endereços para Ponteiros

Com endereçamento indireto de memória, o endereço (local da memória) é encontrado em um endereço de 16-bit ou 32-bit. Este endereço pode ser guardado em uma das seguintes áreas:

- **Memória M:** como operando endereçado absolutamente ou como variável endereçada simbolicamente via tabela de símbolos.
- **Pilha de Dados Local:** como operando endereçado absolutamente ou como variável temporária declarada na lista de declarações dos blocos.
- **DBs Globais:** como operando endereçado absolutamente. Quando DBs compartilhados (Globais) são usados para guardar ponteiros, deve-se tomar o cuidado de abrir o DB correto (p.ex. *OPN DBn*) antes de ser acessado.
- **DBs Instance:** como operando endereçado absolutamente. Quando dados instance são usados, os seguintes pontos devem ser observados:

**OBs e funções:** Dentro de funções ou OBs, um ponteiro que está armazenado num DB instance, pode ser usado exatamente como se estivesse armazenado num DB global. Somente deve-se lembrar que ao invés do registrador DB, agora o registrador DI é que está sendo usado.

**FBs:** Dentro de FBs, dados instance, que são parâmetros ou variáveis estáticas, geralmente não podem ser usados simbolicamente para endereçamento indireto de memória.

Acesso absoluto a dados locais dentro de um FB é, em princípio, possível através do "endereço" apresentado na tabela de declarações, contudo, deve ser observado que, quando a FB é usada como multi instance, o endereço não é o endereço absoluto especificado no instance DB. O endereço de fato estará no AR2.

### Nota

Quando você passa ponteiros por memória de endereçamento indireto para blocos ou quer manter o valor permanentemente em variáveis estáticas, então você tem que copiar o valor de ponteiro do parâmetro ou variável de estática em uma variável temporária e então tem que completar o acesso por esta variável temporária.

## Exemplo de Endereçamento Indireto

FC30: Exemplo de endereçamento indireto

Network 1: abrir DB com endereçamento indireto

```
L   #dbnumber           // Cópia o número do DB no MW100
T   MW 100              //
OPN DB[MW 100]          // Abre o DB
```

Network 2: Loop de apagamento

```
L   P#18.0              // Endereço final(DBW18)como Ponteiro
T   MD 40               // no MD 40;
L   10                  // Preseta contador do loop em 10
anf: T MB 50             // e transfere para MB 50;
L   0                   // Carrega valor inicial '0'
T   DBW[MD 40]          // e transfere para o DB;
L   MD 40               // Carrega o ponteiro e
L   P#2.0               // decrementa de 2 Bytes
-D                      // e então transfere de volta
T   MD 40               // para MD 40;
L   MB 50               // Carrega o contador do loop
LOOP anf                // decrementa e se necessário salta;
```

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.9



Conhecimento em Automação  
Training Center

### Descrição

Este exemplo mostra uma função que inicializa os dados de um DB com o valor "0". O nº do DB é passado para a função por um parâmetro de entrada.

O DB endereçado é antes de tudo aberto no segmento 1. Por isso, o nº do DB passado (parâmetro: #dbnumber) é copiado numa memory word (MW100) e então o DB é aberto através desta memory word.

No segmento 2, as primeiras 10 words do DB são setadas em "0" pelo loop. O loop usa a instrução LOOP, de maneira que o contador de loop é memorizado no MB 50.

A transferência do valor "0" para cada DBW do DB aconteceu com o auxílio do endereçamento indireto via MD40.

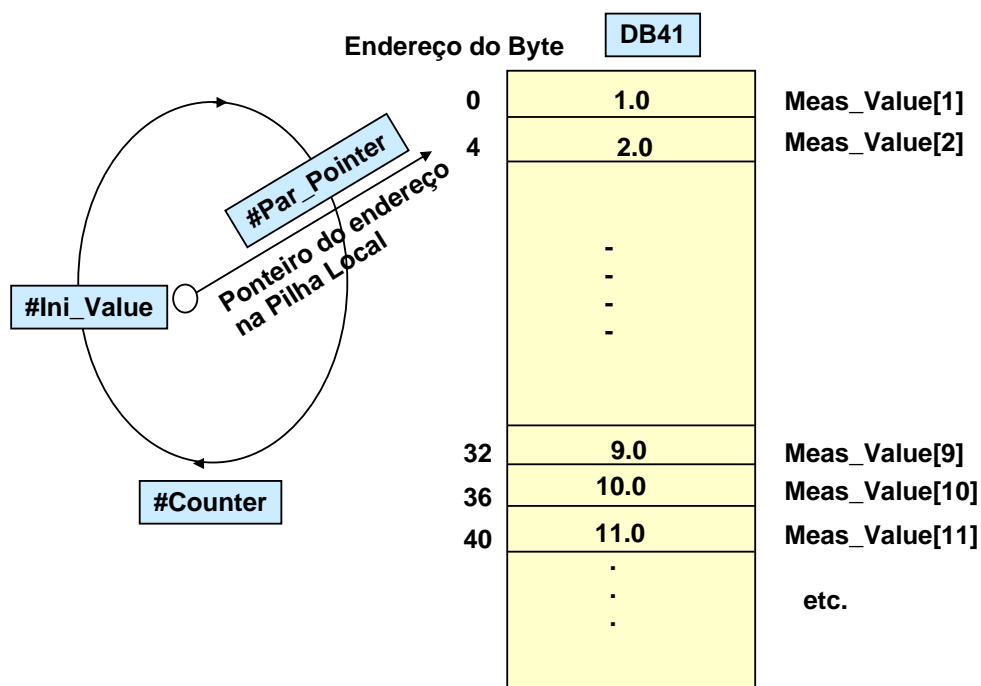
Antes de entrar no loop, o ponteiro com o endereço do último DBW (DBW 18) é carregado no MD 40. A cada loop o endereço de acesso no MD40 é decrementado de P#2.0, pois os endereços são zerados de word em word e não de byte em byte no DB.

### Notas

Na prática, também faria sentido programar o endereço inicial e o comprimento da área com "0" como parametrizáveis e checar antes de abrir o DB, se o DB realmente existe e com o comprimento necessário.

Ao invés de se utilizar o endereçamento indireto via ponteiro com memória M, qual seria uma alternativa melhor? Porquê?

## Exercício 4.1: Programando Loop com Endereçamento Indireto



SIMATIC S7  
Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.10



Conhecimento em Automação  
Training Center

### Objetivo

Familiarizar-se com endereçamento indireto em loop com um exemplo prático.

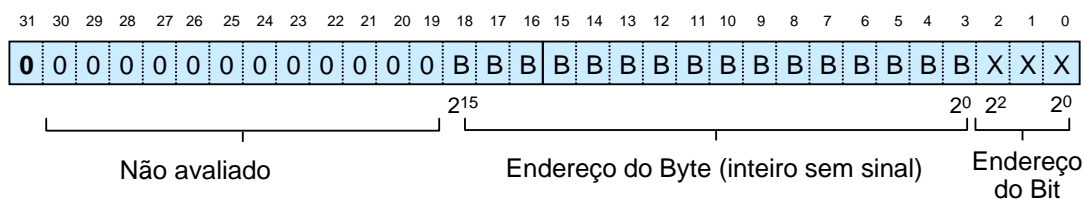
### Definição

Endereçamento indireto de memória é usado para programação de um loop. Com isto, 100 sucessivas células de memória são preenchidas em ordem ascendente com os valores de 1.0 a 100.0.

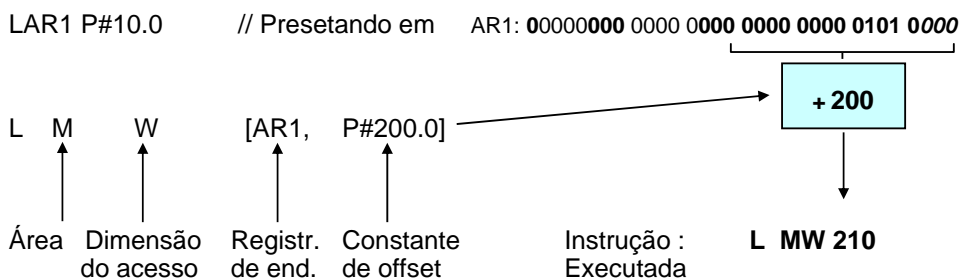
1. Criar um FC41 e um DB41.
2. Nas declarações do DB41, defina a variável *#Meas\_Value* como sendo do tipo:  
ARRAY[1..100] como componente tipo REAL.
3. Na tabela de declarações do FC41, defina um parâmetro de entrada *#DB\_Num* do tipo INT e quatro variáveis temporárias:  
*#L\_Counter* do tipo INT,  
*#Ini\_Value* do tipo REAL,  
*#I\_DB\_Num* do tipo WORD,  
*#Par\_Ponteiro* do tipo DWORD.
4. Dentro do FC41, abra o DB, cuja numeração será passada usando *#DB\_Num*. Use a variável temporária *#I\_DB\_NUM* para isto.
5. Então presete os campos *#Meas\_Value[1]* a *#Meas\_Value[100]* no DB41 em ordem ascendente com os números 1.0 to 100.0.  
Use a programação do loop programming para isto (Instrução: LOOP):
  - Salve o contador de varreduras do loop na variável *#L\_Counter* e o valor de inicialização para os componentes individuais do *Meas\_Value[...]* na variável *#Ini\_Value*.
  - Use o endereçamento indireto de memória para o endereçamento individual dos componentes *#Meas\_Value[...]*. Salve o endereço do acesso na variável *#Par\_Ponteiro*.
6. Chame o FC41 no OB1 e atribua o parâmetro de entrada *#DB\_Num* adequadamente. Transfira os blocos para a CPU e teste o programa.

## Endereçamento Indireto de Registrador de Área Interna

### ❑ Ponteiro de Área interna em AR 1 ou AR2:



### ❑ Sintaxe de Comando :



## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.11



Conhecimento em Automação  
Training Center

### Visão Geral

Com Endereçamento Indireto de Registrador com Área Interna, o endereço (localização da memória) do operando que se está acessando estará em um dos dois registradores de memória (AR1, AR2).

O conteúdo dos registradores de memória serão neste caso um ponteiro de 32-bit com configuração e significado idêntico a memória de endereçamento indireto.

### Sintaxe

Com Endereçamento Indireto de Registrador, as instruções consistem em:

- uma instrução p.ex.: A, L, T, etc.
- um identificador de endereço (I, MB, QD, etc.), que é a combinação do identificador de área (I, Q, M, DB, DI, etc.) e um identificador de dimensão (B=Byte, W=WORD, D=DWORD).
- e a declaração de um AR, que junto com uma constante de offset deve ser colocado dentro de colchetes. Este offset é adicionado ao conteúdo do endereço especificado no AR antes da instrução ser executada.

O conteúdo do registrador de endereços (AR) e o offset formarão o ponteiro de área interna que consiste do endereço de byte + endereço de bit.

A declaração do offset na sintaxe do comando é imperativa.

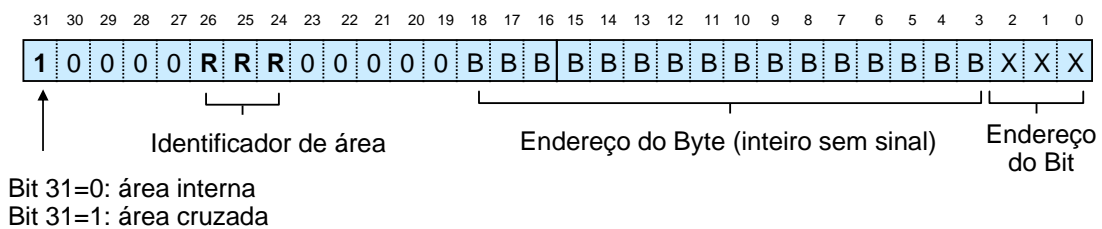
### Notas

- Para endereçamento indireto de byte, word ou double o word, o offset tem que ter no endereço do bit o valor "0", caso contrário um erro de runtime é ativado na CPU durante a execução da instrução.
  - Se o AR1 ou AR2 especificado no registro indireto, contém um ponteiro de área cruzada (veja próxima página), então o identificador de área do ponteiro não será avaliado durante a execução da instrução.
- O identificador de área no identificador de endereço é válido.



## Endereçamento Indireto de Registrador de Área Cruzada

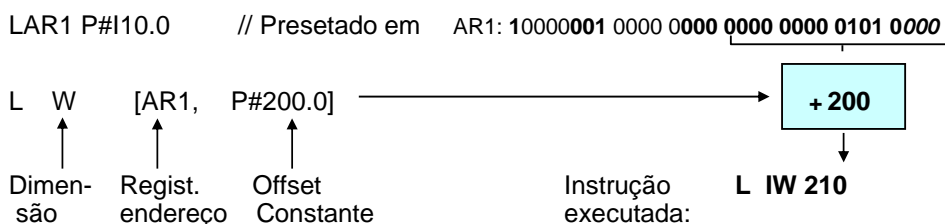
### ❑ Ponteiro de Área cruzada no AR 1 ou AR2:



### ❑ Identificador de Área :

000	I/O	001	Entradas (inputs) (PII)
010	Saídas (outputs) (PIQ)	011	Memórias M
100	Dados no DB Register	101	Dados no DB Register 2 (DI)
110	Dados locais próprios	111	LD do bloco chamado

### ❑ Sintaxe do Comando :



SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.12Conhecimento em Automação  
Training Center

### Visão Geral

No endereçamento indireto de registrador com área cruzada, o identificador de área (I, Q, M, etc.) e endereço (localização de memória) (byte.bit) do operando que se deseja acessar, está no ponteiro de área cruzada em um dos dois registradores de endereços (AR1, AR2).

### Sintaxe

No endereçamento indireto de registrador com área cruzada, a instrução inteira consiste em:

- uma instrução (p.ex.: A, L, T, etc.)
- um identificador de dimensão (B=BYTE, W=WORD, D=DWORD).
- e a declaração de um AR, que junto com uma constante de offset deve ser colocado dentro de colchetes.

O conteúdo do AR neste caso deve ser ponteiro de área cruzada com identificador de área e endereço de byte.bit.

O offset (byte.bit) é adicionado ao conteúdo do endereço especificado no AR antes da instrução ser executada.

A declaração do offset na sintaxe do comando é imperativa.

### Notas

- Para endereçamento indireto de byte, word ou double o word, o offset tem que ter no endereço do bit o valor "0", caso contrário um erro de runtime é ativado na CPU durante a execução da instrução.
- Acesso a dado local próprio (identificador: 110) não é possível com endereçamento indireto de área cruzada. A mensagem de erro "unknown area identifier" é apresentada.

Acesso a dado local próprio só é possível com endereçamento de área interna.

## Instruções para Carregamento dos Registradores de Endereço

### ❑ Carregando o Registrador de Endereços

- LARn (n =1 ou 2): Carregar conteúdo do ACCU1 no ARn
- LARn <Endereço> Carregar conteúdo do <Endereço> no ARn
- LARn P#<Endereço> Carregar endereço do <Endereço> no ARn

#### <Endereço>:

- Registro de processador: AR1, AR2 (p.ex. *LAR1 AR2* e *LAR2 AR1*)
- Variáveis de 32-bit como: MDn, LDn, DBDn, DIDn (p.ex. *L DBD5*, etc.)
- variáveis simból. 32-bit : (compartilhada e local) variável compartilhada 32-bit (p.ex. *LAR1 "Index"*, etc.) e variáveis TEMP de OBs, FBs e FCs (p.ex. *LAR1 #Address*, etc.)

#### P#<Endereço>

- Ponteiro para endereços booleanos absolutos: In.m, Qn.m, Mn.m, Ln.m, DBDn.m, DIDn.m (p.ex. *LAR1 P#M5.3*, *LAR2 P#I3.6*, etc.)
- Ponteiro para local, endereços simbólicos  
OB: variáveis TEMP (p.ex.: *LAR1 P##Pointer*, etc.)  
FB: variáveis IN, OUT, INOUT, STAT e TEMP  
FC: variáveis TEMP (*LAR1 P##Loop*, etc.)

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.13



Conhecimento em Automação  
Training Center

### Carregando Operandos

Com ajuda de funções de carga, é possível iniciar o registrador de endereços com valores definidos.

A função de carga LARn (n=1, 2) carrega o ponteiro no ARn. ACCU1 ou ARn ou uma double word da área de endereços de memória M, dado local temporário, dado de DB global e dado de DB instance podem ser usados como fonte. O acesso pode ser absoluto ou simbólico.

Se você não especificar o endereço, o conteúdo do ACCU1 é automaticamente carregado no ARn. O conteúdo do registrador carregado ou a double word tem que corresponder ao formato do ponteiro da área.

### Carregando Ponteiros

Ponteiros diretos (endereços) podem naturalmente também ser carregados em endereços nos registradores de endereços.

Com auxílio da instrução:

- L P#<identificador de área>n.m

você pode carregar um ponteiro de área cruzada diretamente no AR especificado. Somente um acesso absoluto é possível.

Um ponteiro de área cruzada para uma variável local de nome *#Address* pode por exemplo com ajuda da seguinte instrução:

- LARn P##*Address* (n=1, 2)

ser carregado em um dos dois AR. O ponteiro de área cruzada formado contém o endereço do primeiro byte da variável local.

Este acesso é possível com todas variáveis TEMP dos OBs, FBs e FCs, assim como em variáveis IN, OUT, INOUT e variáveis STAT dos FBs.

### Nota

Se você quer carregar ponteiros em parâmetros IN, OUT e INOUT (*#Param*) de uma FC nos ARs, isto não é possível fazer de maneira direta. Um passo intermediário deve ser feito:

- L P##*Param* (Carrega o parâmetro ponteiro *#Param* no ACCU1)
- LARn* (Carrega o conteúdo do ACCU1 no ARn)

## Outras Instruções com Registradores de Endereço

### ❑ Transferindo de um Registro de Endereço

- TAR<sub>n</sub> (n =1 or 2): Transferindo o conteúdo do AR<sub>n</sub> para ACCU1
- TAR<sub>n</sub> <Endereço> Transferindo o conteúdo do AR<sub>n</sub> para o <Endereço>

<Endereço>:

- Registro de processador: AR2 (p.ex. *TAR1 AR2* )
- Variáveis abs. de 32-bit: MD<sub>n</sub>, LD<sub>n</sub>, DBD<sub>n</sub>, DID<sub>n</sub> (p.ex. *TAR2 MD5*, etc.)
- variáveis simból. 32-bit : variável compartilhada 32-bit (p.ex. *TAR1 "Index"*, etc.) e variáveis TEMP de OBs, FBs e FCs (p.ex. *TAR1 #Address*, etc.)

### ❑ Troca dos Registros de Endereço

- CAR Troca dos conteúdos entre AR1 e AR2

### ❑ Adicionando ao Registro de Endereço

- +AR<sub>n</sub> Soma ACCU1-L ao AR<sub>n</sub>
- +AR<sub>n</sub> P#n.m Ad. ponteiro de área interna P#n.m no AR1 ou AR2

#### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.14



Conhecimento em Automação  
Training Center

#### Transferindo do AR

A instrução TAR<sub>n</sub> transfere a completa área do ponteiro do registro de endereço AR<sub>n</sub>. O outro registro de endereço ou uma double word da área de memória M, dado local temporário, dados de DB ou DI podem ser especificados como destino.

Se nenhum endereço for especificado, TAR<sub>n</sub> transfere o conteúdo do AR para o ACCU1. O valor anterior do ACCU1 é deslocado para o ACCU2; o conteúdo do ACCU2 é perdido.

O conteúdo do ACCU3 e ACCU4 (S7-400) se mantém inalterados.

#### Troca entre ARs

A instrução CAR troca os conteúdos de AR1 e AR2.

#### Somando aos ARs

Um valor pode ser somado aos registros de endereços, por exemplo, incrementar o valor do endereço em todo ciclo de loop de um programa . O valor pode ser especificado como uma constante (ponteiro de área-interna) com a instrução ou como o conteúdo da palavra direita do ACCU1-L .

As instruções +AR1 e +AR2 interpretam o valor encontrado no ACCU1 como um numero em formato INT, expande-o para 24 bits com o sinal correto e soma-o ao endereço contido no registrador. Desta forma o ponteiro pode ser menor. Se o valor ficar abaixo ou acima da área permitida para endereços de byte (0 ... 65 535), nenhum impacto será causado; os bits acima simplesmente são desprezados.

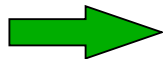
A instrução +AR<sub>n</sub> P#n.m adiciona um pointer de área interna ao registro de endereço especificado. O ponteiro da área especificada desta forma pode ter um tamanho máximo de P#4095.7.

Nenhuma das instruções especificadas acima ou na página anterior modificam os bits da status word.

## Características do Registrador de Endereçamento Indireto

### ❑ Uso interno do AR1 pelo editor STL/LAD/FBD

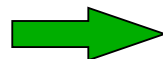
- Acessando os parâmetros de FCs, o **AR1 register** e o **DB register** são sobrescritos, se os parâmetros forem do tipo complexo ("ARRAY", "STRUCT", "DATE\_AND\_TIME").
- Acessando parâmetros INOUT de FBs, **AR1 register** e o **DB register** são sobrescritos, se o parâmetro INOUT é do tipo complexo ("ARRAY", "STRUCT", "DATE\_AND\_TIME").



**Nenhum acesso a parâmetro local deve acontecer entre o carregamento do registro de endereço e o registro de acesso indireto da variável desejada.**

### ❑ Uso interno do AR2 pelo editor STL/LAD/FBD

- O **AR2 register** e o **DI register** são usados como registrador básico de endereços para endereçamento de todos parâmetros e variáveis STAT de **FBs**.



**Se AR2 ou DI forem sobrescritos pelo usuário em um FB, nenhum acesso aos parâmetros do FB ou variáveis STAT pode acontecer depois disso, quer dizer, sem uma restauração de ambos os registros.**

- Nenhuma restrição com relação ao **AR2** e **DI register** dentro de FCs

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.15



Conhecimento em Automação  
Training Center

### Registrador de Endereços AR1

O Editor STEP 7 usa o registrador de endereço AR1 para acessar os parâmetros de bloco complexos. Dentro de funções, com todos acessos simbólicos de parâmetros do tipo "ARRAY" ou "STRUCT", os registros AR1 e DB são sobrescritos.

Bem como, com acessos de parâmetros in/out do tipo "ARRAY" ou "STRUCT" dentro de um FB, os registros AR1 e DB são sobrescritos.

Acesso simbólico de variáveis temporárias em FBs ou FCs não sobrescrevem nem o AR1 e nem o registrador DB.

### Registrador de Endereços AR2

O Editor STEP 7 usa a área interna de registro de endereçamento indireto para acesso simbólico de dados instance, quer dizer, de todos os parâmetros e das variáveis estáticas de um FB. O registrador DI especifica o respectivo DB instance e o AR2 o respectivo multi instance dentro do instance DB.

Nenhum acesso a dados instance pode acontecer depois destes registradores DI e AR2 serem sobrescritos, pois o conteúdo destes dois registradores não são restabelecidos. Se você quer usar o AR2 ou registrador DI em um FB para seus próprios propósitos, então o seguinte procedimento é recomendado:

1. Salve o conteúdo do DI e AR2 em variáveis do tipo DWORD:

```
TAR2 #AR2_REG // Salva AR2 numa variável temporária #AR2_REG
L DINO        // Carrega o conteúdo do DI no ACCU1
T #DI_REG      // Salva numa variável temporária #DI_REG
```

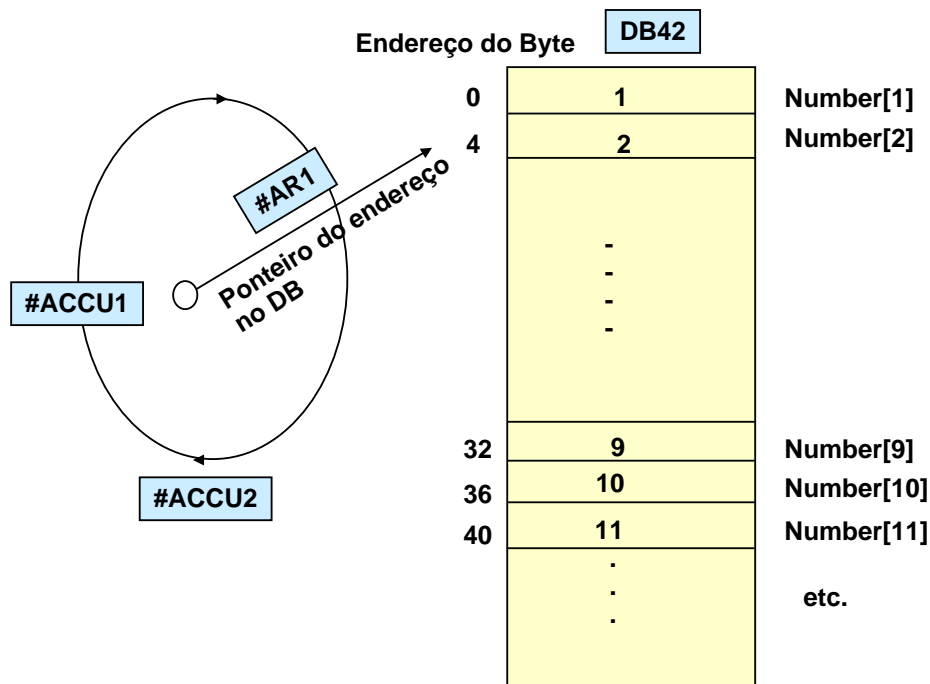
2. Use o registrador DI e AR2 para seus próprios propósitos. Nenhum acesso a parâmetros de FB's ou variáveis estáticas podem ocorrer durante este segmento.

3. Restabeleça o registrador DI e o AR2:

```
LAR2 #AR2_REG // Carrega AR2 com conteúdo do #AR2_REG
OPN DI[#DI_REG] // Restabelece o registrador DI
```

Os parâmetros do FB e variáveis estáticas podem novamente serem acessados simbolicamente.

## Exercício 4.2: Loop com Registrador de Endereçamento Indireto



SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.16Conhecimento em Automação  
Training Center

### Objetivo

Familiarizar-se com o uso de endereçamento indireto de registrador em Loop com um exemplo prático.

### Definição

Endereçamento indireto do registrador é usado para programação de um loop. Os valores de 1 a 100 são escritos sucessivamente em 100 memórias.

Programar uma solução ótima (sem variáveis temporárias adicionais) do exercício 4.1.

Salve os valores do contador de Loop e inicialização nos acumuladores.

Para endereçar os componentes Tank[.] use o AR1 (endereçamento indireto do registrador com área interna).

### Procedimento

1. Gerar uma FC42 e um DB42.
2. Nas declarações do DB42, defina uma variável *#Number* do tipo ARRAY[1..100] com os componentes tipo DINT.
3. Na tabela de declarações do FC42, defina um parâmetro de entrada *#DB\_Num* do tipo WORD e uma variável temporária *#I\_DB\_Num* do tipo WORD.
4. Dentro do FC42, primeiro abra o DB, cujo número será passado usando *#DB\_Num*. Use a variável temporária *#I\_DB\_Num* para isto.
5. Então presete os campos *#Number[1]* a *#Number[100]* no DB42 em ordem ascendente com os números 1.0 to 100.0.
  - Use programação de loop para isto (Instrução: LOOP):
  - Use endereçamento indireto de registro com AR1 para endereçamento individual dos componentes dos *#Number[.]*.
6. Chamar FC42 no OB1 e atribua parâmetro para o parâmetro de entrada *#DB\_Num* adequadamente. Transfira os blocos para CPU e teste o seu programa.

## Tipos de Ponteiros do STEP 7

- ❑ **Ponteiro 16-bit para Endereçamento Indireto de Memória**
  - Para acesso indireto de memória de temporizadores, contadores e DBs abertos
- ❑ **Ponteiro 32-bit p/ Endereçamento Indireto de Memória e Registrador**
  - Ponteiro 32 bit de área interna para acesso indireto de memória e registrador de endereços em PI, PQ, I, Q, M, DB, DI e L (pilha local)
  - Ponteiro 32 bit de área cruzada para acesso indireto do registrador de endereços PI, PQ, I, Q, M, DB, DI, L e V (Pilha de dados locais do bloco chamado)
- ❑ **Ponteiro 48-bit (Tipo de Dado POINTER)**
  - Tipo de dado próprio para passagem de parâmetros para blocos (FBs e FCs)
  - Contém, em adição ao ponteiro 32-bit de área-cruzada, a declaração do número do DB
- ❑ **Ponteiro 80-bit (Tipo de Dado ANY)**
  - Tipo de dado próprio para passagem de parâmetros para blocos (FBs e FCs)
  - Contém, em adição ao ponteiro 32-bit de área-cruzada, a declaração do número do DB, tipo de dado e fator de repetição

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.17



Conhecimento em Automação  
Training Center

### Tipos de Ponteiros do STEP 7

Além dos tipos de ponteiros descritos na seção anterior (16-bit, 32-bit área-interna e 32-bit área-cruzada), o STEP 7 reconhece dois tipos de ponteiros adicionais:

- Ponteiro 48-bit (tipo de dado "POINTER")
- Ponteiro 80-bit (tipo de dado "ANY")

Os pointers de 16 e 32-bit podem ser carregados diretamente no acumulador ou registrador de endereço e assim podem ser usados para endereçamento indireto dentro dos blocos.

Os pointers tipo POINTER e ANY (maior que 32 bit) não podem ser carregados diretamente nos registradores e usados para endereçamento indireto nos blocos. Eles são usados exclusivamente para um endereçamento completo dos parâmetros atuais na passagem para parâmetros formais dos blocos chamados.

Por exemplo, você pode declarar um parâmetro tipo POINTER ou ANY num bloco e durante a chamada do bloco atribuir a este parâmetro o endereço atual.

### POINTER

O dado tipo POINTER é usado principalmente pelo STL/LAD/FBD Editor para passar um parâmetro atual do tipo complexo, como "ARRAY", "STRUCT", e "DT", para chamada de FB ou FC.

O editor STL/LAD/FBD checa imediatamente a consistência do tipo de dado e o comprimento atribuído ao parâmetro atual, basta somente passar o endereço inicial completo para o parâmetro atual interior.

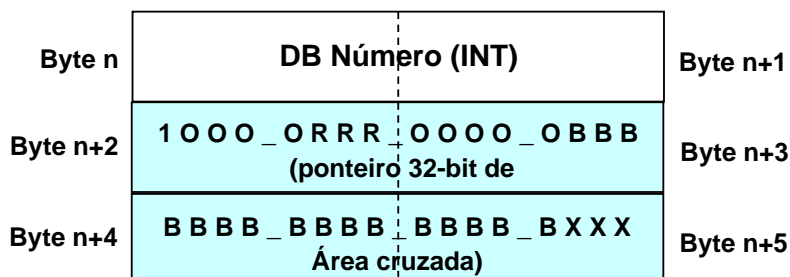
Dentro de um bloco chamado, você pode então acessar os parâmetros atuais do registrador indireto usando este PONTEIRO.

### ANY

O pointer tipo ANY é principalmente usado pelo STEP7 para atribuir parâmetros as funções de sistema (SFCs) e blocos de funções de sistema (SFBs). Parâmetros dos tipos de dados ANY podem também serem utilizados pelo usuário para gerar blocos mais poderosos.

## Estrutura e Atributos de um Dado Tipo "POINTER"

### □ Estrutura do tipo de dado: "POINTER" (PONTEIRO)



### □ Atributos de um parâmetro tipo "POINTER"

#### ○ Ponteiro mostrado

P#DBn.DBX x.y

P#DIn.DIX x.y

P#Zx.y

com: n= DB número, x= byte-número, y= bit-número

(p.ex.: P#DB5.DBX3.4, P#DI12.DIX10.0, etc.)

com: Z= área, p.ex.: P, I, Q, M e L

(p.ex.: P#I5.3, P#M10.0, etc.)

#### ○ Declaração de Endereço:

MD30

#Motor\_on

"Motor\_1".speed

(Neste caso, identificador do número do DB e endereço do bit são automaticamente fornecidos ao "POINTER")

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.18



Conhecimento em Automação  
Training Center

### Dado tipo POINTER

Um parâmetro do tipo POINTER contém adicionalmente ao pointer de área cruzada, um nº de DB (inteiro sem sinal - faixa: 0 ... 65,535). Em todos os demais casos, quando o endereço endereçado se encontra em outras áreas (P, I, Q, M, L), "0" é colocado nos primeiros dois bytes do "POINTER".

### Atributo de Parâmetro

Se durante a chamada de um bloco (FC ou FB), um parâmetro tipo POINTER tem que ser definido, isto pode ser feito via apresentação do pointer ou por declaração do endereço.

### Apresentação do Pointer

Neste caso, um pointer (P#...) tem que ser declarado desde o primeiro bit do endereço, como a seguir:

- P#DB10.DBX2.0 // Data bit 2.0 in DB10, identificador de área "DB"
- P#I5.3 // I5.3, DB número = 0, identificador de área "PII"

### Declaração do Endereço

Neste caso, a declaração do endereço é suficiente (sem P#...). O endereço pode ser declarado absolutamente, ou seja, via nº de DB, identificador de endereço e endereço de byte ou bit associado, como a seguir:

- DB5.DBW10 // Bit 10.0, DB número = 5, identificador de área // "DB" ou simbólico.
- #Motor\_on, "Motor\_1".speed

Em ambos casos, o editor STL/LAD/FBD estabelece o número de DB associado, o identificador de área e o endereço de byte.bit e coloca isto no "POINTER".



## Configuração de um dado tipo "ANY"

### □ Ponteiro "ANY" para Tipos de Dados

Byte n	16#10	Tipo de Dado
Byte n+2	Fator de repetição	
Byte n+4	DB Número	
Byte n+6	1 0 0 0 _ O R R R	O O O O _ O B B B
Byte n+8	B B B B _ B B B B	B B B B _ B X X X

### Tipo de Dado      Identificador

VOID	00
BOOL	01
BYTE	02
CHAR	03
WORD	04
INT	05
DWORD	06
DINT	07
REAL	08
DATE	09
TOD	0A
TIME	0B
S5TIME	0C
DT	0E
STRING	13

### □ Ponteiro "ANY" para Tipos de Parâmetros

Byte n	16#10	Tipo de Parâmetro
Byte n+2	16#0001	
Byte n+4	16#0000	
Byte n+6	16#0000	
Byte n+8	Número do Temporizador, Contador ou Bloco	

### Tipo Parâmetro      Identificador

BLOCK_FB	17
BLOCK_FC	18
BLOCK_DB	19
BLOCK_SDB	1A
COUNTER	1C
TIMER	1D

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.19Conhecimento em Automação  
Training Center

### Tipo de Dado ANY

O pointer tipo ANY contém adicionalmente ao pointer de cruzamento de área e o n° de DB, um identificador de tipo de dado um fator de repetição. Com isto, é possível identificar não somente um endereço individual mas também uma área completa de dados.

Há duas versões do ponteiro ANY:

- para variáveis com tipos de dados: O ponteiro ANY então contém um identificador de sintaxe (syntax-ID) 16#10 para STL, um identificador para o tipo de dado, um fator de repetição, o número do DB e um ponteiro de área cruzada.
- para variáveis com tipo de parâmetros: neste caso, o pointer ANY consiste meramente do identificador de sintaxe (syntax-ID) 16#10 para STL, um identificador de tipo de parâmetro e um número sem sinal de 16-bit no byte n+8 e byte n+9, que reflete o número do bloco. Os bytes n+4, ..., n+7 são preenchidos com "0".

### Declaração dos Ponteiros ANY

Variáveis do tipo ANY geralmente podem ser declaradas como parâmetros IN, OUT e INOUT em FCs e FBs.

A declaração oferece uma possibilidade adicional como variável temporal dentro de FBs. Com ajuda desta variável temporária é possível criar um ponteiro ANY que é mutável durante a execução do programa e passá-lo para uma chamada de bloco (ver: Atribuição de Parâmetros Indiretos do Tipo ANY).

### Identificador de Área (RRR):

000	I/O	001	Entradas (inputs) (PII)
010	Saídas (outputs) (PIQ)	011	Memórias M (Bit memory)
100	Dados no DB register	101	Dados no DI register
110	Dados locais próprios	111	LD do bloco chamado



## Atributos dos Parâmetros de Dados Tipo "ANY"

### □ Apresentação do Pointer:

#### ○ P#[Data block.]Bit address Type Number

P#DB10.DBX12.0 REAL 20      Pointer na área do DB10, iniciando com Byte 12, constituído de 20 endereços do dado tipo REAL (ARRAY[1..20] OF REAL)

P#I 10.0 BOOL 8      Pointer de um campo de 8 bits no IB10

### □ Declaração de Endereços:

#### ○ absoluto:

DB5.DBD10

Data type: DWORD, Repetition factor: 1  
DB number: 5, Pointer: P#DB5.DBX10.0

IW32

Type: WORD, RF: 1, DB-No: 0, Pointer: P#I32.0

T35

Type: TIMER, No.: 35

#### ○ simbólico:

#Motor\_1.speed

"Pump:Start"

com dado tipo elementar, o compilador estabelece o correto tipo de dado, fator de repetição 1 e pointer

### □ Nota

com atributo simbólico (ARRAY, STRUCT, STRING, UDT), o identificador de tipo de dado 02 (BYTE) e a dimensão da área em bytes é somente estabelecida pelo compilador e introduzida no ponteiro ANY.

SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.20



Conhecimento em Automação  
Training Center

### Atributos

Um parâmetro tipo "ANY" pode ser declarado como uma declaração direta de endereços (variáveis).

### Apresentação do Ponteiro

Declarando-se o pointer display (p.ex.: P#DB5.DBX10.0 INT 8) o Editor

STL/LAD/FBD monta um ponteiro ANY que corresponde em tipo e em número com as declarações.

Atribuições na apresentação do ponteiro sempre fazem sentido, quando uma área de dados está sendo endereçada, para a qual nenhuma variável tenha sido definida ou por exemplo, nenhuma variável adequada (p.ex.: ARRAY ou STRUCT) pode ser definida (p.ex.: P, PII, PIQ, M).

Adicionalmente a exibição de ponteiro absoluto deve ser usada quando dentro do bloco chamado a informação correta sobre o fator de repetição e o tipo de dados é requerido (p.ex.: ARRAY[1..8] de REAL).

### Apresentação de endereço

Um parâmetro do tipo "ANY" pode também ser diretamente definido com o endereço para o qual o ponteiro ANY será apontado. Esta declaração pode ser absoluta ou pelo nome de variável simbólica.

Com a declaração do endereço absoluto o editor STL/LAD/FBD estabelece automaticamente o tipo de dado associado (BOOL, BYTE, WORD, DWORD), um fator de repetição "1", o número do DB assim como ponteiro de área cruzada no primeiro bit do endereço e entra com estes valores na estrutura do ponteiro.

Igualmente, o Editor STL/LAD/FBD estabelece a informação correta pelo endereço quando a declaração acontecer pelo nome simbólico e a variável fornecida é do tipo de dados elementar.

### Nota

Se a variável é do tipo complexo (p.ex. ARRAY[1..8] de REAL), então o Editor STL/LAD/FBD meramente coloca as informações em bytes sobre a área ocupada pela variável (p.ex.: fator de repetição: 32, dado tipo: BYTE)

## Atributos Indiretos de Parâmetros do Tipo “ANY”

### ❑ Atributos através dos parâmetros atuais temporários dos tipos de dados ANY

- declare variável temporária do tipo de dado ANY na chamada do bloco

p.ex.: temp aux\_pointer ANY

- carregue a variável temporária ANY com a informação do ponteiro

p.ex.:

```
LAR1 P##aux_pointer      // Carrega endereço ponteiro aux.
L    B#16#01             // Carrega identificador B#16#01
T    LB [AR1,P#0.0]      // e transfere para Offset 0
L    ...
...
```

- Defina o parâmetro de bloco tipo ANY (Targetfield) com a variável

ponteiro auxiliar (aux\_pointer)

p.ex.:

```
CALL FC    111
      Targetfield:=#aux_pointer
```

### ❑ Vantagem

- redefinição do parâmetro do pointer ANY durante execução do programa.

## SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.21



Conhecimento em Automação  
Training Center

### Definição Indireta

O bloco chamado também pode designar um parâmetro de FC ou FB de dados tipo ANY com uma variável temporária de dados tipo ANY. Esta variável temporária é armazenada na pilha de dados locais do bloco de chamada.

Neste caso, o Editor de STL não passa nenhum ponteiro à variável temporária (na pilha de dados local), mas assume que esta variável temporária ANY já contém o ponteiro da variável de fato desejada.

O Editor passa neste caso o ponteiro ANY contido na variável temporária da FC ou da FB chamado.

### Vantagem

Você tem a possibilidade de setar um ponteiro ANY em um parâmetro ANY que você pode mudar durante execução do programa. O ponteiro variável ANY ser muito útil, especialmente nos casos das funções do sistema, como SFC 20 (BLKMOV) ou SFC 21 (FILL) por exemplo.

## Avaliando um ponteiro tipo "ANY"

Address	Declaration	Name	Type	Initial Value	Comment
0.0	in	Par_Pointer	ANY		
	out				
	in_out				
0.0	temp	Data_type	BYTE		
2.0	temp	WF	WORD		
4.0	temp	DB_Nr	WORD		
6.0	temp	Area_Pointer	DWORD		

Network 1: Estabelecimento do tipo de dado, fator de repetição, número do DB e ponteiro de área

```

L   P##Par_Pointer      // Carrega endereço do #Pointer no ACCU1
LAR1                                // e daqui carrega no AR1;
L   B [AR1,P#1.0]       // Cria o tipo de dado do ponteiro
T   #Data_type          // e carrega em variável temporária
L   W [AR1,P#2.0]       // Cria o fator de repetição
T   WF                  // e carrega em variável temporária
L   W [AR1,P#4.0]       // Cria o número do DB
T   #DB_Nr              // e carrega em variável temporária
L   D [AR1,P#6.0]       // Cria o ponteiro de área
T   #Area_Pointer       // e carrega em variável temporária

```

### SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.22



Conhecimento em Automação  
Training Center

### Visão Geral

Dentro do bloco chamado (FB ou FC) as informações que estão no ponteiro passado do tipo "POINTER " ou "ANY " podem ser lidas e avaliadas adequadamente com ajuda de registrador de endereçamento indireto.

### Procedimento

A avaliação da informação passada para um pointer "ANY " está listada nos passos seguintes. Os passos estão relacionados ao exemplo anterior onde um parâmetro de entrada (tipo "ANY ") com o nome `#Par_Pointer` e várias variáveis temporárias para o armazenamento temporário de informação foram declaradas.

1. Em primeiro lugar, um ponteiro de área cruzada é estabelecido na passagem do ponteiro "ANY" e carregado no registro de endereço AR1. Isto acontece com a instrução:

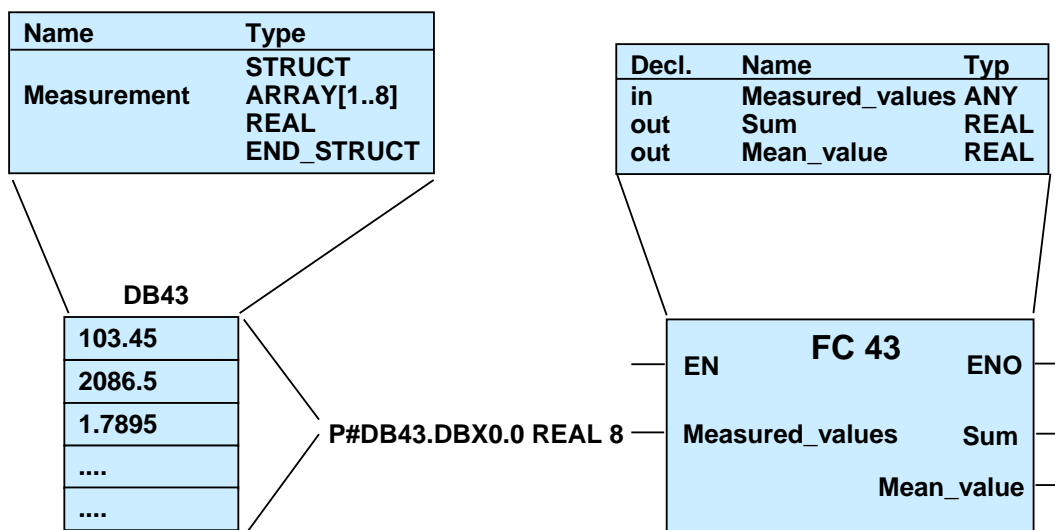
- `LAR1 P# #Par_Pointer` // em FBs ou
- `L P# #Par_Pointer` // em FCs, o endereço deve antes de tudo ser  
`LAR1` // carregado no ACCU1 e dele para o  
// registrador AR1

O ponteiro "ANY" passado é, no caso de um FB, armazenado no DB instance (este é automaticamente aberto) ou com uma FC, na pilha de dados locais do bloco que originou a chamada.

2. Usando o Registrador de endereçamento indireto, as informações passadas no ponteiro "ANY" podem ser lidas agora e, por exemplo, pode ser armazenado temporariamente em variáveis temporais do bloco para outros processamentos.

- `L B[AR1,P#1.0]` // lê o identificador do tipo de dados do parâmetro  
// atual no ACCU1
- `L W[AR1,P#2.0]` // lê o fator de repetição no ACCU1
- `L W[AR1,P#4.0]` // lê o nº do DB no ACCU1, ou "0" quando o  
//parâmetro atual está armazenado em P, PII, PIQ, // M, L.
- `L D[AR1,P#6.0]` // lê ponteiro de área cruzada do parâmetro atual  
// no ACCU1

## Exercício 4.3: Função para calcular valor da soma e média



SIMATIC S7

Siemens AG 1998. All rights reserved.

Date: 4/10/2007  
File: PRO2\_04P.23Conhecimento em Automação  
Training Center

### Visão Geral

FCs ou FBs genéricas podem ser geradas com ajuda do dado tipo "ANY". Não são comprometidos FCs ou FBs genéricas de tipo de dados específicos. Elas podem "adaptar-se" dinamicamente para os tipos de dados ou comprimentos de campo passados a eles, ou comprimentos de campo passados a eles.

### Objetivo

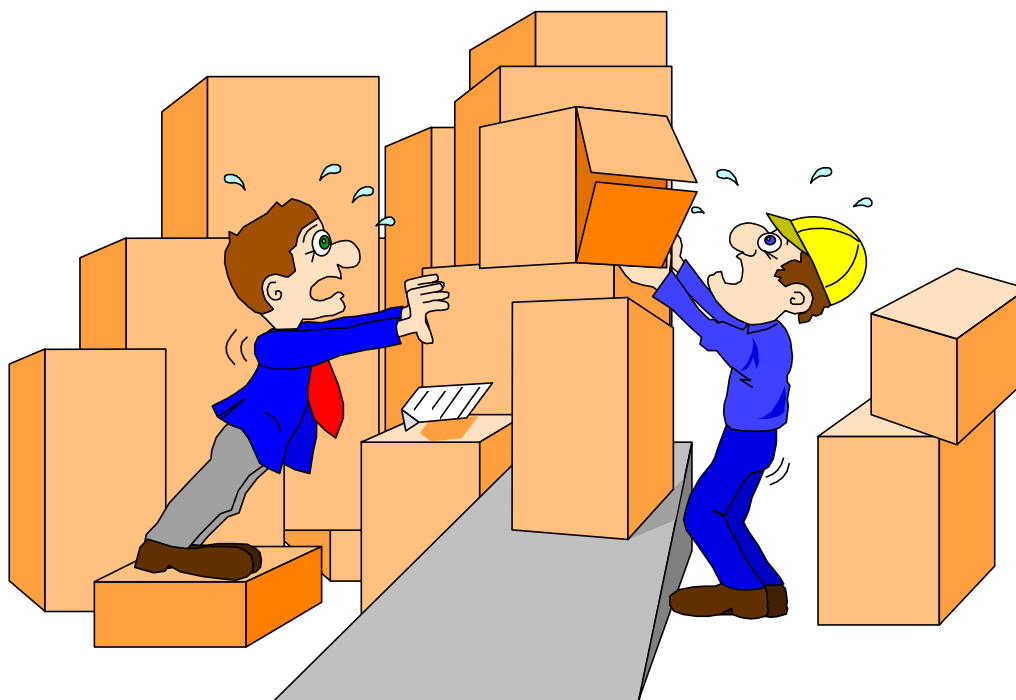
Gerar a FC43 com a seguinte funcionalidade:

- A função espera um campo REAL valores com parâmetro de entrada *Measured\_values* (tipo "ANY").
- A função fornece os valores somados dos elementos de campo passados no parâmetro *#Sum* (tipo: REAL) e o valor médio de todos os elementos de campo no parâmetro de saída *#Mean\_value* (tipo: REAL).
- Se outro tipo de dado for passado, ocorrerá um erro (parâmetro ENO, ou seja, BR bit=0, nº REAL inválido para *#Sum* e *#Mean\_value*

### Procedimento

1. Crie a FC43 e declare os parâmetros entrada e de saída listados anteriormente. Também declare as correspondentes variáveis temporárias para armazenamento temporário de informação sobre fator de repetição, DB n. e o ponteiro de área do parâmetro atual.
2. Começar com a leitura do ponteiro "ANY" passado dos dados tipo identificador e sair do FC43 adequadamente, se o tipo de dados do parâmetro atual não é REAL.
3. Em um loop (instrução de LOOP), programe a soma de todos os elementos de campo. Calcule a soma e valor médio e atribua os resultados aos parâmetros de saída correspondentes.
4. Gerar o DB43. Declare uma variável *Measurement* do tipo ARRAY[1..8] OF REAL em DB43 e entre valores apropriados.
5. Programe a chamada de FC43 em OB1. Nomeie o parâmetro de entrada no pointer display. Nomeie endereços na área de memória para os parâmetros de saída
6. Descarregue os blocos participantes na CPU e teste o resultado.

## Tipos de Dados e Variáveis STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.1Conhecimento em Automação  
Training Center

### Conteúdo

	Pág.
Significado das Variáveis e Tipos de Dados .....	2
Propriedades e Declaração de Variáveis .....	3
Vista Geral dos Tipos de Dados no STEP 7 .....	4
Tipos de Dados Elementares no STEP 7 .....	5
Importância dos Tipos de Dados Complexos .....	6
Tipos de Dados Complexos no STEP 7 .....	7
Tipos de Parâmetros no STEP 7 .....	8
Áreas para gravação de Variáveis .....	9
Funcionamento da metodologia da Pilha de Dados Local .....	10
Exemplo: Utilização como memória para rascunho .....	11
Blocos de Dados (DB) .....	12
Tipo de Dado: ARRAY .....	13
Declaração e Inicialização dos ARRAYS .....	14
Armazenagem das Variáveis ARRAY na Memória .....	15
Tipo de Dado: STRUCT .....	16
Declaração dos STRUCTs .....	17
Armazenagem das Variáveis STRUCT na Memória .....	18
Tipos de Dados Definido pelo Usuário: UDTs .....	19
Uso dos UDTs .....	20
Tipo de Dado: DATE_AND_TIME .....	21
Funções para processamento de Variáveis DT .....	22
Tipo de Dado: STRING .....	23
Armazenagem das Variáveis STRING na Memória .....	24
Funções para processamento de Variáveis STRING .....	25
Exercício 5.1: Uso dos Tipos de Dados Complexos .....	26
Exercício 5.2: Acessando Tipos de Dados Complexos .....	27
Exercício Adicional 5.3: Lendo "Time-of-Day" com SFC 1 (READ_CLK) .....	28

## Significado das Variáveis e Tipos de Dados

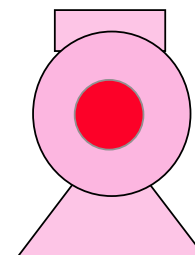
### Tipos de dados caracterizam as propriedades básicas do dado

- Área contínua: p.ex. Velocidade atual
- Propriedade "sim/não": p.ex. distúrbio

### Os tipos de dados definem:

- A faixa de valores permitidos (INT: -32 368 ... +32 367, etc.)
- As instruções possíveis (instruções aritméticas: +, -, etc.)
- Tipos de dados abstratos da representação subordinada aos bits na memória

### As Variáveis permitem que você salve e mais tarde continue a processar valores



Actual\_speed: REAL

Set\_speed: REAL

Disturbance: BOOL

Enable: BOOL

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.2



Conhecimento em Automação  
Training Center

### Vista Geral

Nos modernos sistemas computadorizados que foram desenvolvidos para simplificar e aumentar a velocidade do processamento de cálculos complicados. A capacitação para processar grandes quantidades de informação, armazenar e tornar novamente acessível uma extensa relação de aplicações.

A informação disponível ao controlador consiste de uma selecionada quantidade de dados sobre o "mundo real". Os dados representam uma abstração da realidade porque propriedades casuais e sem importância dos objetos envolvidos não são levados em conta para este problema específico.

### Tipos de Dados

É freqüente a dificuldade para decidir como os dados serão representados. A escolha é freqüentemente restrita as possibilidades disponíveis. De outra forma, as propriedades dos objetos os quais são descritos por dados devem ser corretamente refletidos, ou de outra forma, as instruções que são necessárias para o gerenciamento do processo devem também serem capazes de utilizar os dados corretamente.

O tipo de dado determina quais valores são aceitos pelo dado e quais instruções podem ser executadas com estes valores.

Os tipos de dados definem unicamente:

- a faixa de valores permissíveis
- as instruções possíveis

Os tipos de dados abstratos igualmente sob representação subordinada (formato) de bits individuais, como eles, são finalmente guardados na memória do computador.

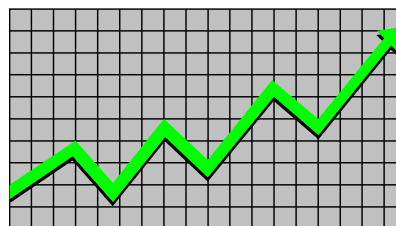
### Significado das Variáveis

Junto aos comandos, as variáveis são os mais importantes elementos do sistema de programação. Sua tarefa é salvar valores em um programa o qual poderá ser processado posteriormente. O valor da variável pode ser salvo em "qualquer lugar" da memória do PLC.

## Propriedades e Declaração de Variáveis

As seguintes propriedades são determinadas pela declaração da variável:

- Nome simbólico
- Tipo de dado
- Faixa de validade



Meas\_point: ARRAY[1..10]

Meas\_point[1]: Real

Meas\_point[2]: Real

Meas\_point[3]: Real

Meas\_point[10]: Real

Variáveis podem ser declaradas:

- Na tabela de símbolos global (tipos de dados elementares)
- Na tabela de declarações de blocos de dados globais (todos os tipos de dados)
- Na tabela de declarações de um bloco lógico (OB, FB e FC)

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.3



Conhecimento em Automação  
Training Center

### Variáveis "Convencionais"

Na convencional de PLC, os endereços da memória do PLC são acessados diretamente pela especificação da área de memória (p.ex.: M= Memórias M, I=Entradas, etc.), a dimensão do acesso (p.ex.: B=byte, W=palavra (word), etc.) e pela especificação do endereço do byte/(bit). Estas áreas endereçáveis de memória utilizam endereços que podem ser usadas dentro de um programa com diferentes propósitos, por exemplo, como inteiro (p.ex.: DINT), como um número em ponto flutuante (isto é, REAL) ou simplesmente como uma coleção de sinais individuais (p.ex.: WORD).

Até agora era necessário que o programador se lembrasse do formato e a localização das memórias individuais para propósitos de aplicação. Falhas de programa poderiam ocorrer facilmente, porque endereços errados de memória ou formatos incorretos eram não intencionalmente usadas nas instruções.

### Declaração de Variáveis

Posteriormente sistemas PLC (p.ex.: STEP 5) permitiram o uso de símbolos para facilitar a leitura de programas. STEP 7 foi um passo adiante e utiliza variáveis em vez de endereços e símbolos no PLC.

Para explicitar a declaração de uma variável, as seguintes propriedades são fixadas:

- Nome simbólico de variáveis
- Tipo de dados de variáveis
- Faixa de validade

Se variáveis são declaradas, o editor de programa pode então, por exemplo, usar informações de tipos de dados para checar a permissibilidade das instruções como os parâmetros atribuídos na chamada de blocos.

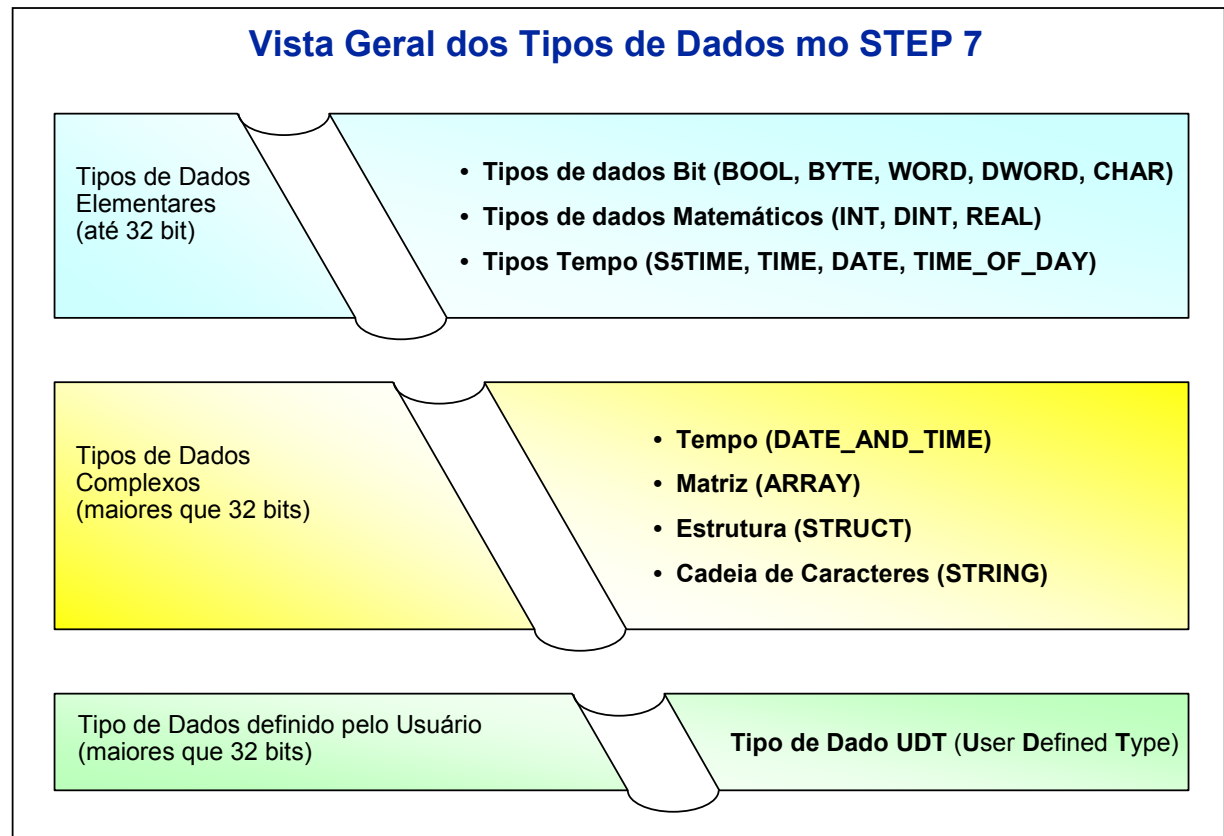
### Faixa de Validade

Variáveis que são declaradas na tabela de símbolos globais ou em DBs globais podem ser endereçados por todos os blocos na pasta de programa. Estas variáveis são chamadas variáveis globais por esta razão.

Variáveis e parâmetros que são declaradas na seção de declaração de um bloco lógico são chamados locais; eles podem somente usados dentro da seção de declaração do mesmo bloco.



## Vista Geral dos Tipos de Dados no STEP 7



SIMATIC S7  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.4



Conhecimento em Automação  
Training Center

### Vista Geral

A solução de tarefas de automação usados em sistemas computadorizados é baseado em algoritmos, em cujo processo os dados são coletados por sensores, de forma a colocar na saída novos valores para os atuadores. Programas são basicamente formas de algoritmos que dependem de representação de dados específicos ou estrutura de dados.

### Tipos de Dados Elementares

Os tipos de dados elementares reúnem os "átomos" de cada sistema de programação.

A escolha dos tipos de dados elementares de um sistema de programação diz muito a respeito da área de aplicação pretendida.

Em STEP 7, os tipos de dados elementares são predefinidas em acordo com IEC 1131-3. Os tipos de dados são escolhidos em função do tipo de tarefa a ser executada como processamento binário e processamento de sinais analógicos, um simples sistema de sinalização bem como gerenciamento tarefas de tempo podem ser operadas desta forma.

Com tipos de dados elementares, o tipo de dado determina a quantidade de espaço de memória que uma variável requer. Tipos de dados elementares nunca maiores que 32 bits de comprimento no STEP 7 e pode ser carregada nos acumuladores por inteiro e processadas com instruções STEP 7.

### Tipos de Dados Complexos

A idéia básica de estruturação de dados é diferenciado entre estruturas elementares e superiores. O "former" são os átomos dos quais os tipos de dados são configurados.

No STEP 7, tipos de dados complexos podem somente se usados em conjunto com variáveis declaradas em DBs globais ou em pilha de dados locais. Tipos de dados complexos não podem se completamente carregados em um acumulador e processados.

### Tipos de Dados Definido pelo Usuário

Tipos de dados complexos não tem seu próprio identificador para o tipo de dado e como um resultado não pode ser repetidamente usado para declaração de parâmetro ou variável. Com a ajuda de tipo de dados definidos pelo usuário (UDT), único, tipos de dados estruturados podem ser criados as quais podem então ser usadas tão freqüente como requerido para outras declarações de variáveis ou parâmetros.



## Tipos de Dados Elementares no STEP 7

Palavra-chave	Dimensão (em bits)	Exemplo de uma constante deste tipo
BOOL BYTE WORD DWORD CHAR	1 8 16 32 8	1 or 0 B#16#A9 W#16#12AF DW#16#ADAC1EF5 'w'
S5TIME	16	S5T#5s_200ms
INT DINT REAL	16 32 32	123 65539 or L#-1 1.2 or 34.5E-12
TIME DATE TIME-OF-DAY	32 16 32	T#2D_1H_3M_45S_12MS D#1999-06-14 TOD#12:23:45.12

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.5Conhecimento em Automação  
Training Center

#### BOOL, BYTE, WORD DWORD, CHAR

Variáveis do tipo de dado BOOL consiste de um bit, variáveis de tipos de dados BYTE, WORD, DWORD são seqüências de 8, 16 e 32 bits respectivamente. Os bits individuais não são avaliados neste tipo de dado.

Formas especiais destes tipos de dados são números BCD e o valor de contagem usado em conjunto com a função de contagem, bem como o tipo de dado CHAR, os quais representam um caractere em código ASCII.

#### S5TIME

Variáveis do tipo de dado S5TIME são requeridas para valores específicos de temporização em funções de temporização (S5 funções de temporização). Você especifica o tempo em horas, minutos, segundos ou milissegundos. Você pode entrar com valores de tempo com um underline (1h\_4m) ou sem um underline (1h4m).

As funções FC33 e FC40 da biblioteca convertem S5TIME para formato TIME e o formato TIME para S5TIME.

#### INT, DINT, REAL

Variáveis deste tipo de dados representam números os quais podem ser usados em operações matemáticas.

#### TIME

Uma variável do tipo de dado TIME é formada por uma palavra dupla. Esta variável é usada, por exemplo, por valores de tempo específico em funções de temporização IEC.

Os conteúdos das variáveis são interpretadas como um número DINT em milissegundos e pode ser positivo ou negativo (p.ex.: T#-1s=L#-1 000, T#24d20h31m23s647msw = L#214748647).

#### DATE

Uma variável do tipo de dado DATE é armazenada em uma palavra no formato de um inteiro não sinalizado. Os conteúdos da variável representa o número de dias desde 01.01.1990 (p.ex.: D#2168-12-31 = W#16#FF62).

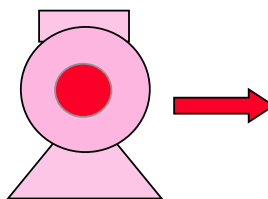
#### TIME\_OF\_DAY

Uma variável do tipo de dado TIME\_OF\_DAY é formada por uma palavra dupla. Ela contém o número de milissegundos desde o início do dia (0:00 horas) na forma de um número inteiro não sinalizado (p.ex.: TOD#23:59:59.999 = DW#16#0526\_5B77).

## Importância dos Tipos de Dados Complexos

### "Melhor" estruturação de Dado:

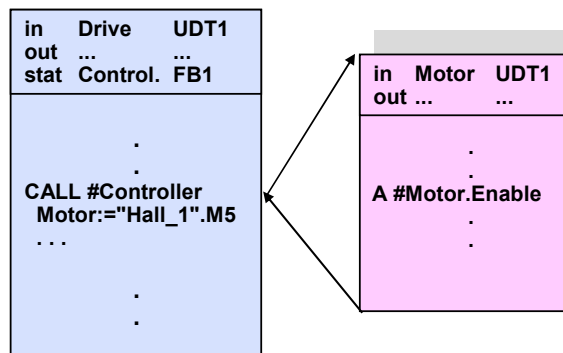
- Adaptado para a tarefa
- com "correto" tipo de dado



Motor: STRUCT	
Set_speed:	REAL
Actual_speed:	REAL
Enable:	BOOL
Disturbance:	BOOL
END_STRUCT	

### Forma Compacta de Dado passado na chamada em um Bloco:

- "muitos" itens de dados podem ser passados em um parâmetro
- Possibilidade de fazer programação estruturada
- blocos "comunicam" somente via barramento de parâmetros
- software reutilizável



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.6Conhecimento em Automação  
Training Center

### Tipos de dados Complexos

Tipos de dados complexos (arrays e estruturas) resultam de agrupamento de dados elementares ou tipos de dados complexos.

Tipos de dados complexos são usados para organização de dados complexos. Neste caminho, o programador pode gerar tipos de dados para atender a tarefa particular. Ele pode combinar unidades de informações elementares logicamente relacionadas com uma nova "unidade" que possua seu próprio nome.

Um típico exemplo para uma estrutura é o arquivo de dados para um drive. O drive é descrito como um arquivo de atributos (propriedades, estados), como *#Set\_speed*, *#Actual\_speed*, *#Enable* and *#Disturbance*. Muitos destes atributos podem ser na forma de estruturas.

Um *#Disturbance* pode, por exemplo, ser construído por componentes individuais (bits) os quais constituam mais informações exatas para o usuário a respeito da causa do distúrbio.

### Programação Estruturada

Dados complexos, em particular, podem ser passados em uma chamada de um bloco como uma unidade, isto é, em um parâmetro do bloco chamado. Desta forma, uma multiplicidade de unidades de informações elementares podem ser transferidas entre a chamada e o bloco chamado de modo elegante e compacto.

### Software Reutilizável

Este tipo de transferência de dados possibilita a criação de programação estruturada e garante um alto grau de reutilização de software criado uma única vez.

A tarefa a ser automatizada é dividida em blocos individuais. Na listagem do bloco chamado, nenhum acesso a endereços globais, como Memórias M ou variáveis em DBs globais são executadas. O processamento da informação é executada exclusivamente com parâmetros, nos quais dados relevantes do processo são passados.

Os resultados do processamento são retornados nos parâmetros do bloco chamado.

## Tipos de Dados Complexos no STEP 7

Palavra chave	Dimensão (em bits)	Exemplo	
<b>DATE_AND_TIME</b> (Data e Horário)	64	DT#99-06-14-12:14:55.0	
<b>STRING</b> (Caracter "string" com máx. 254 caracteres)	8 * (número de caracteres +2)	'Isto é uma string' 'SIEMENS'	
<b>ARRAY</b> (Grupo de elementos do mesmo tipo de dado)	definido pelo usuário	Meas_vals: ARRAY[1..20] INT	
<b>STRUCT</b> (Estrutura, Grupo de elementos de diferentes tipos de dados)	definido pelo usuário	Motor: STRUCT Speed : INT Current : REAL END_STRUCT	
<b>UDT</b> (User Defined Data Type = "Template" constituído de tipos de dados elementares e/ou complexos)	definido pelo usuário	UDT como bloco	UDT como elemento array
		STRUCT Speed : INT Current : REAL END_STRUCT	Drive: ARRAY[1..4] UDT1

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.7Conhecimento em Automação  
Training Center

### Arrays e Estruturas

Com auxílio de ARRAYS (Field - campos), diversos objetos do mesmo tipo podem ser combinados dentro de um tipo de dado. Um ARRAY é um tipo de dado que consiste de um número fixo de elementos de um tipo de dado. Um índice é designado para cada elemento de um ARRAY. O índice é usado para acessar o elemento.

Um exemplo de um ARRAY é uma série de medições, a qual é constituída de um número fixo de valores de medição individual.

Da mesma forma que um ARRAY permite a combinação de elementos de mesmo tipo em um conjunto, os tipos de dados STRUCT (estrutura) habilitam a união de elementos de diferentes tipos de dados.

Tipos de dados complexos são pré definidos. Os tipos de dados DATE\_AND\_TIME tem uma dimensão de 64 bits. A dimensão dos tipos de dados ARRAY, STRUCT e STRING são definidas pelo usuário.

### Tipo de dado definido pelo usuário

Com a ajuda dos tipos de dados definidos pelo usuário (UDT), você pode definir tipos de dados especiais (estruturas) que podem ser então usadas tão freqüentemente quanto você queira na declaração de parâmetros e variáveis.

A estrutura de dados é armazenada em um bloco UDT (UDT1 ... UDT65535) e pode ser usado – como um "template" – na declaração do tipo de dado de uma variável ou um parâmetro em OBs, FCs, FBs e DBs.

Com a ajuda de UDTs, você pode salvar tempo de digitação uma vez que a mesma estrutura é solicitada diversas vezes.

Exemplo: Você requer a mesma estrutura 10 vezes em um bloco de dados. Primeiro você define a estrutura e a salva como UDT1, por exemplo.

No DB, você define a variável "Drive" como um array com 10 elementos do tipo UDT1:

```
Drive: array[1..10]  
      UDT 1
```

Então, você criou faixas de 10 dados com a estrutura definida em UDT 1 sem digitação adicional.

## Tipos de Parâmetros no STEP 7

Palavra chave	Dimensão (em bits)	Exemplo
TIMER	16	Contact time: TIMER · SI #Contact_time
COUNTER	16	NoCompParts: COUNTER · LC #No_Comp_Parts
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	16	Recall: BLOCK_FB · UC #Recall
Pointer	48	Measure: POINTER · L P##Measure
ANY	80	Measured Values: ANY · L P##Meas_Values

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.8Conhecimento em Automação  
Training Center

### Tipos de Parâmetros

Adicionalmente aos tipos de dados elementares e complexos, você pode definir tipos de parâmetros para os parâmetros dos FCs e FBs. Com este parâmetro formal, você pode então executar as mesmas instruções como com os endereços atuais.

Estes parâmetros formais devem então serem alimentados com parâmetros atuais associados durante a chamada de um bloco.

### TIMER e COUNTER BLOCK\_xx

Estes tipos de parâmetros definem um parâmetro formal do tipo TIMER ou COUNTER.

Com a ajuda dos tipos de parâmetros BLOCK\_FB ou Block\_FC, blocos de programa podem ser passados como parâmetros para chamada de blocos. De qualquer maneira, somente estes blocos (FBs, FCs) os quais não tem controle sobre parâmetros ou variáveis estáticas (BLOCK\_FB) podem ser passados.

Os blocos lógicos formais podem somente ser chamados usando as instruções UC e CC (não CALL) dentro da chamada de bloco.

Não existem restrições para a passagem dos blocos de dados (DB, Sdb) e para as instruções associadas (p.ex.: OPN ...).

### POINTER

POINTER é usado quando tipo de dado *any* pode ser o tipo de dado do parâmetro atual. O POINTER contém o endereço inicial completo (número do DB, área de dado, endereço do byte e endereço do bit) do parâmetro atual.

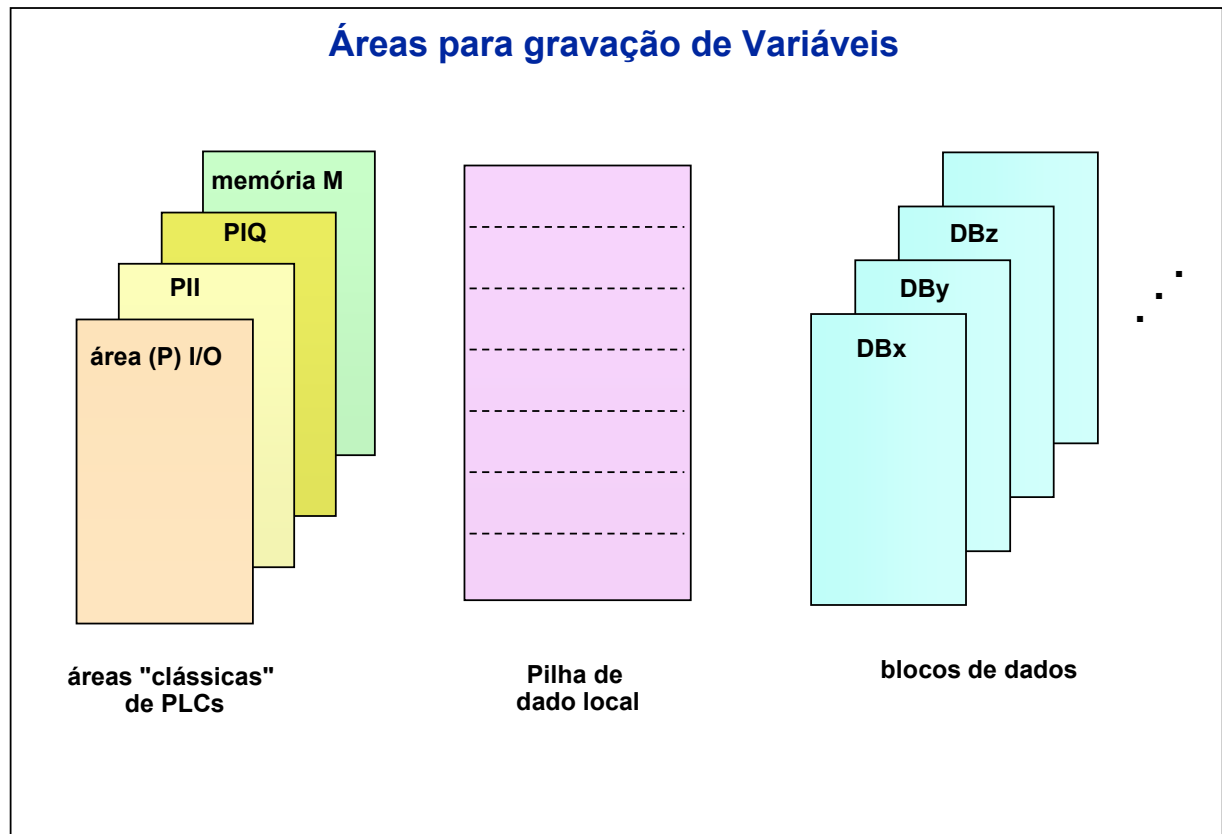
Você pode determinar um parâmetro formal do tipo POINTER pela atribuição do endereço do parâmetro atual, p.ex.: P#M50.0.

### ANY

ANY é usado quando o tipo de dado *any* pode ser o tipo de dado do parâmetro atual. Em adição ao endereço inicial completo, informações sobre o tipo de dado e a dimensão é também passado em um ponteiro tipo ANY.

P#M10.0 Byte 10 (Campo de 10 componentes do tipo de dado BYTE começando com MB 10).

## Áreas para gravação de Variáveis



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.9Conhecimento em Automação  
Training Center

### Vista Geral

A parte dos blocos de programa, um programa do usuário também consiste de dados contendo informações sobre estados do processo, sinais, etc., os quais são então processados de acordo com as instruções no programa do usuário.

Variáveis podem ter uma localização de memória permanente na imagem de processo, área de endereço de memória M ou em DBs ou eles podem ser parametrizados dinamicamente durante a execução do programa na Pilha L.

### PII, PIQ, Memória M I/O

Variáveis elementares podem ser declaradas na tabela de símbolos globais da pasta de programa. Adicionalmente ao nome simbólico da variável, você deve também dar uma área de memória consistindo do identificador de área e dimensão, bem como um tipo de dado (p.ex.: FullCrate MW 10 INT).

Diferentemente da tabela de símbolos no STEP 5 (Assignment List), o editor de programas permite não somente o uso do nome simbólico como também o endereço absoluto. Ele também monitora o uso correto da variável quando parâmetros são atribuídos na chamada de blocos (teste do valor digitado).

As variáveis que são declaradas na tabela de símbolos globais são globais. Todos os blocos na pasta de programa podem acessá-la.

**Pilha de Dados Local** A Pilha de Dados Local (L stack) é uma área para armazenamento:

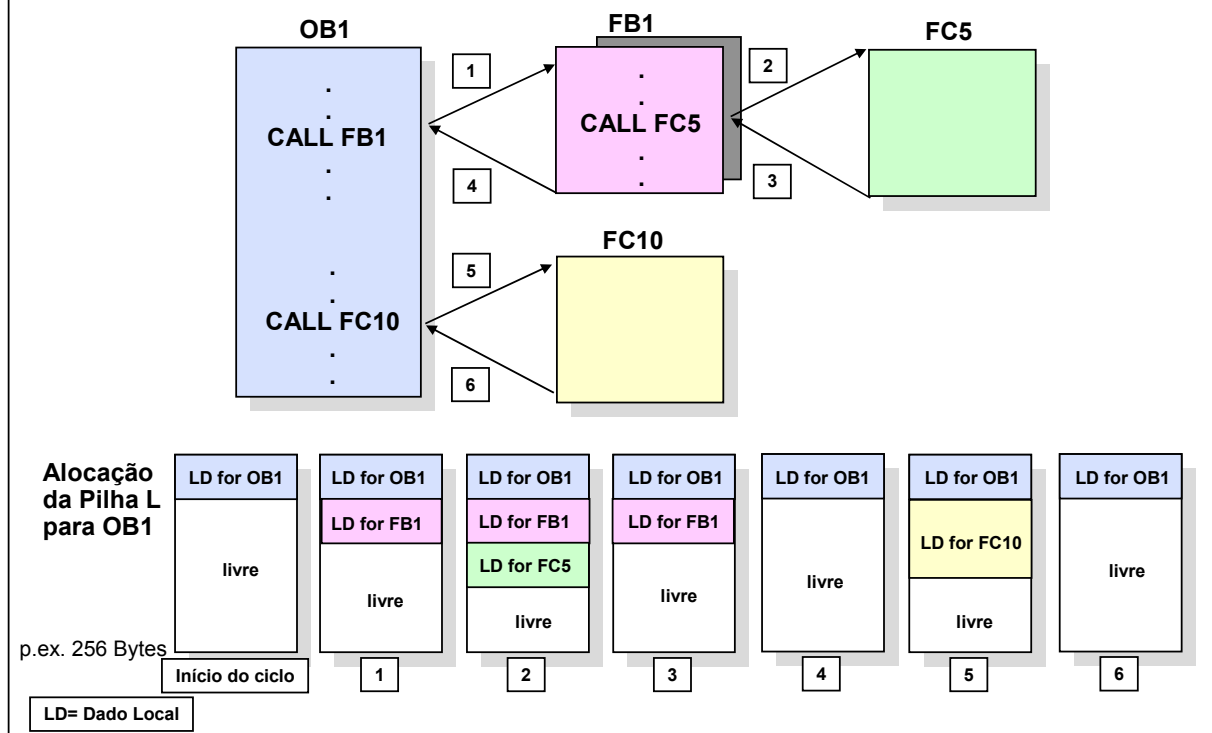
- variáveis temporárias de um bloco lógico, incluindo informações de partida de OB
- parâmetros atuais podem ser passados quando chamando funções
- resultados lógicos intermediários em programas LAD

Áreas na pilha L para variáveis são atribuídas dinamicamente quando o programa é executado no bloco e são habilitadas uma vez novamente depois da execução do bloco.

### Blocos de Dados

DBs são blocos usados pelos blocos lógicos para armazenamento de dados pelo programa do usuário. Diferentemente das variáveis temporárias, o conteúdo das variáveis nos DBs não são sobrescritas quando a execução do bloco é completada.

## Funcionamento da metodologia da Pilha de Dados Local



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.10Conhecimento em Automação  
Training Center

**Pilha de Dados Local** Para cada classe de prioridade, isto é, para cada OB é alocada sua própria Pilha L para variáveis temporárias dos OBs ou dos blocos chamados adicionalmente.

Antes de um bloco (OB, FB ou FC) ser processado, o sistema reserva memória dinâmica na Pilha L para variáveis temporárias declaradas na parte da declaração dos blocos. A memória é habilitada após BE (fim de bloco).

### Seqüência

O slide acima mostra uma seqüência típica de execução cíclica do OB1. Antes do OB1 ser processado, o sistema operacional reserva (aloca) espaço de memória para as variáveis temporárias do OB1. Junto as variáveis temporárias declaradas pelo usuário, uma área de 20 byte é também reservada e inicializada para informações de partida.

- 1 Antes da execução do FB1, o sistema operacional reserva memória para as variáveis temporárias do FB1. A respectiva área de memória é alocada logo em seguida da memória para as variáveis temporárias do OB1.
- 2 Antes da execução do FC5, o sistema operacional reserva memória para as variáveis temporárias do FC5. A respectiva área de memória é alocada logo em seguida da memória para as variáveis temporárias do FC1.
- 3 Após a conclusão do FC5, a memória associada é habilitada novamente.
- 4 Após a conclusão do FB1, a memória associada é habilitada novamente.
- 5 Agora o sistema operacional reserva memória para o FC10. Esta área Now the operating system reserves memory for the FC10. Esta área de memória é alocada logo em seguida da memória para as variáveis temporárias do OB1. Deste modo o espaço exato de memória, que já havia sido usado previamente pelo FB1 e FC5, é sobrescrito pelas variáveis temporárias originais do FB1 e FC5.

### Vantagem

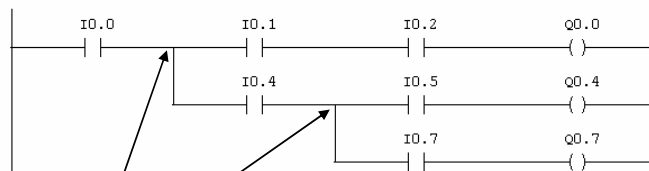
O gerenciamento da memória temporária é realizada pelo sistema operacional e não pelo programa do usuário (erros de programação).

Se a respectiva classe de prioridade é interrompida por um OB com outra classe de prioridade, os dados locais não necessitam serem salvos. Os vários OBs são alocados, neste caso, em sua própria pilha de dados local.

## Exemplo: Utilização como memória para rascunho

### Ramificação em LAD

Network 2: Title:



Locais de ramificação

### Representação STL

Network 2: Title:

```

A      I      0.0
=      L      20.0
A      L      20.0
A      I      0.1
A      I      0.2
=      Q      0.0
A      L      20.0
A      I      0.4
=      L      20.1
A      L      20.1
A      I      0.5
=      Q      0.4
A      L      20.1
A      I      0.7
=      Q      0.7
  
```

Variáveis auxiliares da  
Pilha de Dados Local

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.11



Conhecimento em Automação  
Training Center

#### Ramificação no Editor LAD

O exemplo acima mostra exemplo da representação de ramificação, como ela pode agora ser programada pelo usuário com ajuda do Editor LAD do STEP 7 pela inserção de bobinas de saída adicionais (p.ex. Q 0.7).

#### Memória Rascunho e Conectores

Com STEP 5, a programação de uma ramificação não era diretamente possível. O usuário deve inserir uma variável auxiliar, como regra um bit de memória (conector), para a localização do ponto de ramificação do network. No próximo segmento – um para cada ramificação adicional – esta variável auxiliar é então usada em cada caso como uma entrada e escaneada.

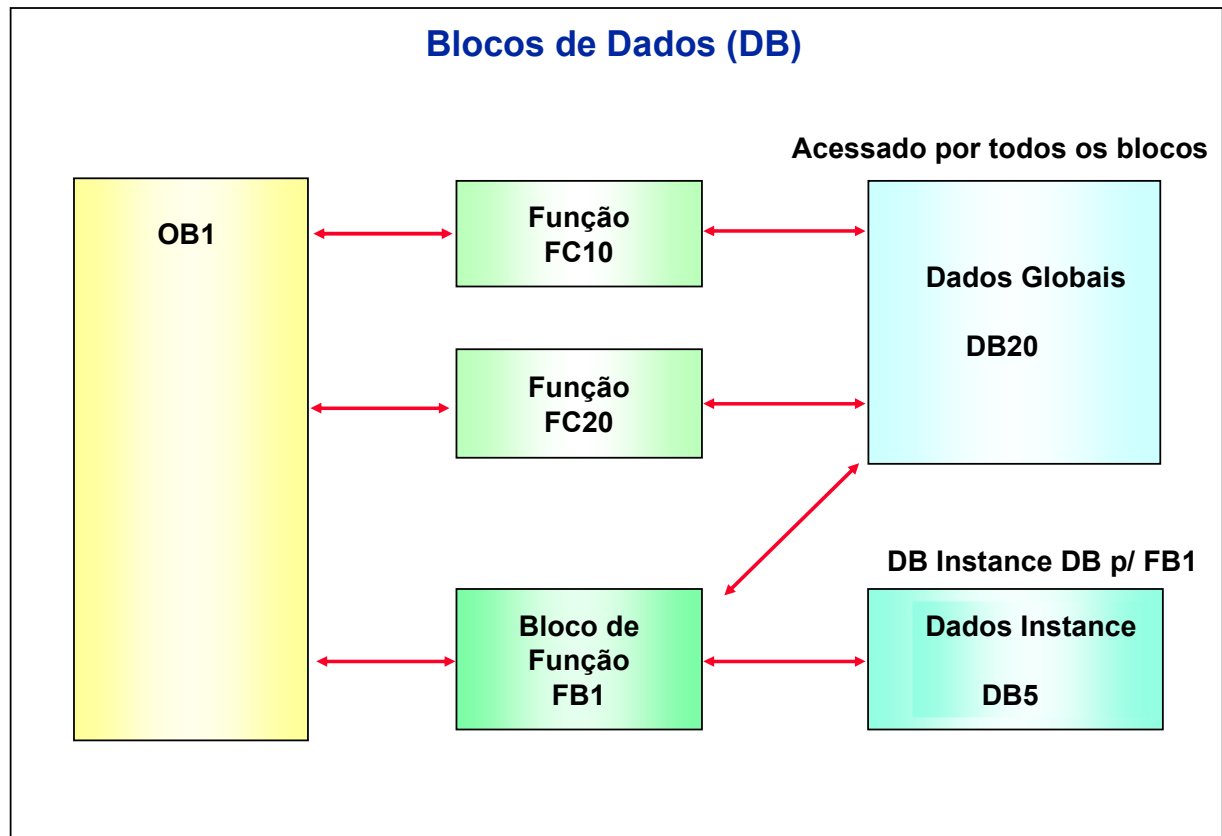
#### Editor LAD

Com a ajuda do Editor LAD, é possível programar as ramificações diretamente com STEP 7. Internamente, uma variável auxiliar – um bit da pilha de dados local – é também usado para localização da ramificação pelo Editor LAD.

O uso de memória na pilha de dados local garante neste caso, que as duas variáveis auxiliares (L 20.0 e L 20.1) não sejam sobrescritas por blocos chamados ao mesmo tempo.

#### Variáveis Temporárias

O usuário pode também declarar variáveis temporárias na parte de declarações e acessar as variáveis absolutamente ou simbolicamente, isto é, através do nome identificador especificado.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.12Conhecimento em Automação  
Training Center**Vista Geral**

DBs são usados para armazenar dados do usuário. Como os blocos lógicos, os DBs ocupam espaço na memória do usuário. Dados variáveis (por exemplo, valores numéricos), com os quais o programa do usuário trabalha, são encontrados no DBs.

O programa do usuário pode acessar dados de um DB via instruções de bit, byte, word (palavra) ou double word (palavra dupla). Os acessos podem ser absolutos ou simbólicos.

**Área de Aplicação**

DBs podem, dependendo de seus conteúdos, ser instalados pelo usuário em diferentes modos. Distinção é feita entre:

- Blocos de Dados Globais (compartilhados): Eles contêm informações que podem ser acessadas por todos os blocos lógicos do programa do usuário.
- Blocos de Dados Instance: Eles são sempre reservados por um FB. Os dados deste DB somente é processado pelo FB associado.

**Criação de DBs**

DBs Globais são cada um criados usando o Editor DB ou de acordo com um pré-determinado "tipo de dado definido pelo usuário".

Blocos de Dados Instance são gerados então quando um bloco FB é chamado.

**Registrador**

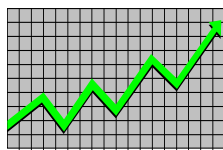
A CPU tem dois registradores de DBs, o registrador DB e o registrador DI. Deste modo dois blocos de dados podem ser abertos ao mesmo tempo.



## Tipo de Dado: ARRAY

### ARRAY (campo):

- Grupo de componentes do mesmo tipo de dados



Meas\_value: ARRAY[1..10]

Meas_value[1]:	Real
Meas_value[2]:	Real
Meas_value[3]:	Real

⋮

Meas_value[10]:	Real
-----------------	------

- Declaração:
  - Uni dimensional:  
*Fieldname*: ARRAY[*minIndex*..*maxindex*] OF *data type*;
  - Multi-dimensional:  
*Fieldname*: ARRAY[*minindex1*..*maxindex1*,*minindex2*..*maxindex2*,...] OF *data type*;  
*Index*: Data type INT (-32768...32767)

### Exemplos:

- Declaração da variável:
  - Uni dimensional: *Meas\_value*: ARRAY[1..10] OF *REAL*;
  - Multi-dimensional: *Position*: ARRAY[1..5,2..8,...] OF *INT*;
- Acesso à uma variável:
  - L #Meas\_value[5]                      // Carrega o quinto elemento do ARRAY
  - // Meas\_value no ACCU1
  - T #Result[10,5]

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.13



Conhecimento em Automação  
Training Center

### Vista Geral

O tipo de dado ARRAY representa um campo com um número fixo de componentes (elementos) do mesmo tipo de dado.

Um campo (=ARRAY) pode ter até 6 dimensões (número de índices). As seguintes restrições se aplicam para os tipos de dados componentes de um array:

- elementares (sem restrição)
- complexos (DATE\_AND\_TIME, STRUCT, UDT)
- tipos sem parâmetros
- sem FBs (modelo multi-instance)

Os ARRAYS não podem ser aninhados. O limite da faixa de índices mínimos e máximos é determinado pela faixa de valores dos INT, isto é, de -32768 a +32767.

### Acesso

Instruções STL podem ser usadas para acessar componentes array dos tipos de dados elementares. Um componente array é endereçado com o nome array e um índice entre colchetes.

O índice deve ser um valor fixado, isto é, um termo constante. Indexação variável durante a execução do programa não é possível em STL..

### Nota

Indexação variável para elementos array individuais somente é possível na linguagem S7-SCL. Acesso variável pode somente ser implementada em STL com a ajuda da memória ou registrador de endereçamento indireto.

## Inicialização e Declaração dos ARRAYS

The screenshot displays two windows from the SIMATIC Manager software. The left window, titled 'DB5 "Declaration View"', shows a table of variable declarations for a data block (DB5). The right window, titled 'DB5 "Data View"', shows the same data block with actual values.

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	sequence	ARRAY[1..10]	5 (2.730000e+002) , 3 (1.000000e+001)
+4.0		REAL	
+40.0	result	ARRAY[1..5,3..7]	10 (
+2.0		INT	
=90.0		END STRUCT	

Address	Name	Type	Initial Value	Actual Value
0.0	sequence[1]	REAL	2.730000e+002	2.730000e+002
4.0	sequence[2]	REAL	2.730000e+002	2.730000e+002
8.0	sequence[3]	REAL	2.730000e+002	2.730000e+002
12.0	sequence[4]	REAL	2.730000e+002	2.730000e+002
16.0	sequence[5]	REAL	2.730000e+002	2.730000e+002
20.0	sequence[6]	REAL	1.000000e+001	1.000000e+001
24.0	sequence[7]	REAL	1.000000e+001	1.000000e+001
28.0	sequence[8]	REAL	1.000000e+001	1.000000e+001
32.0	sequence[9]	REAL	0.000000e+000	1.000000e+001
36.0	sequence[10]	REAL	0.000000e+000	1.000000e+001
40.0	result[1, 3]	INT	5	5
42.0	result[1, 4]	INT	5	5

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.14



Conhecimento em Automação  
Training Center

### Vista Geral

No exemplo acima, duas variáveis do tipo de dado ARRAY são declaradas no DB5 com ajuda do Editor de DBs. A criação de uma nova variável somente é possível na "Declaration view" do DB (*View -> Declaration View*):

- sequence: ARRAY[1..10] OF REAL
- result: ARRAY[1..5,3..7] OF INT

### Inicialização dos ARRAYS

Os componentes array individuais podem ser valores pré atribuídos na declaração (não com parâmetros FC, parâmetros in/out dos FBs ou variáveis temporárias). A inicialização dos tipos de dados valores devem ser compatíveis com o tipo de dado componente.

A inicialização dos valores são introduzidos na coluna "Initial Value", separado por uma vírgula. Se diversos componentes sucessivos com o mesmo valor estão sendo inicializados, um fator de repetição pode ser usado para fazer isto. O fator de repetição é colocado em frente ao valor inicialização que é para ser entrada entre parênteses.

### Exemplo

5 (1.23467E+002) // os próximos 5 componentes são inicializados com o  
// valor 123.467

15 (7,2,3) // os próximos 15 elementos são atribuídos  
// alternativamente com os valores 7, 2 e 3

O resultado da inicialização pode ser checada ou alterada no "Data View" (*View -> Data View*). Se o número da inicialização é menor que o número de componentes, somente o primeiro é pré atribuído e o resto são inicializados com "0".

### Aceitação da Inicialização dos Valores

Se uma nova inicialização de valores são introduzidos na "declaration view", estas mudanças somente serão efetivadas (validação como "actual values") depois de ser executada a função do menu *Edit -> Initialize Data Block*.

A inicialização de valores do ARRAYS na declaração dos parâmetros de entrada ou saída nos FBs são aceitos como valores atuais em um DB instance quando estes são gerados.

## Armazenagem das Variáveis ARRAY na Memória

### Arrays unidimensionais

- Tipo de Dado BOOL

	7	6	5	4	3	2	1	0
Byte n <sup>1)</sup>	8	7	6	5	4	3	2	1
Byte n+1		etc.		12	11	10	9	

- Tipo de Dados BYTE, CHAR

Byte n <sup>1)</sup>	Byte 1
Byte n+1	Byte 2
Byte n+2	Byte 3
	⋮

- Tipo de Dado WORD, DWORD,...

Byte n <sup>1)</sup>	----- Word 1 -----
Byte n+1	
Byte n+2	----- Word 2 -----
Byte n+2	
	⋮

<sup>1)</sup> n = par

### Arrays multidimensionais

- Exemplo.  
ARRAY[1..2,1..3,1..2] OF BYTE

Byte n <sup>1)</sup>	Byte 1.1.1
Byte n+1	Byte 1.1.2
Byte n+2	Byte 1.2.1
⋮	Byte 1.2.2
	Byte 1.3.1
	Byte 1.3.2
	Byte 2.1.1
	Byte 2.1.2
	Byte 2.2.1
	Byte 2.2.2
	Byte 2.3.1
	Byte 2.3.2

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.15



Conhecimento em Automação  
Training Center

### Vista Geral

Conhecimento exato do armazenamento das variáveis ARRAY na memória é então necessária quando, durante a execução do programa, componentes individuais são acessados usando memória ou registrador de endereçamento indireto.

### Armazenamento de Variáveis

Uma variável ARRAY sempre começa no limite de uma palavra, ou seja, em um byte com um endereço par. Uma variável ARRAY ocupa a memória até o próximo limite de palavra. Componentes com tipo de dado BOOL começa no último endereço bit significativo, componentes com tipo de dado BYTE e CHAR no endereço byte significativo. Os componentes individuais são listados em seqüência. Em arrays multidimensionais, os componentes são armazenados linha por linha começando com a primeira dimensão. Uma nova dimensão sempre começa no próximo byte com componentes bit e byte, com componentes de outros tipos de dados sempre na próxima palavra.

### Nota

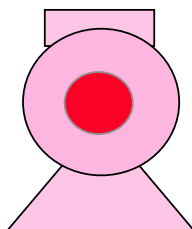
Os endereços dos componentes individuais ARRAY em um DB são mostradas no "Data View" na coluna "Address".

## Tipo de Dado: STRUCT

### STRUCT (Estrutura):

- Grupo de componentes de diferentes tipos de dados
- Declaração:
 

```
StructName:  STRUCT
  Comp1Name: data type;
  Comp2Name: data type;
  ...
END_STRUCT
```



<b>Motor: STRUCT</b>	
<b>Set_Speed:</b>	<b>REAL</b>
<b>Actual_Speed:</b>	<b>REAL</b>
<b>Enable:</b>	<b>BOOL</b>
<b>Disturbance:</b>	<b>BOOL</b>
<b>END_STRUCT</b>	

### Exemplo:

- Declaração de uma variável:
 

```
MotorControl : STRUCT
  ON          : BOOL;
  OFF         : BOOL;
  SetSpeed    : INT;
  ActualSpeed : INT;
END_STRUCT;
```

### Acesso à variável

```
S #MotorControl.ON
L #MotorControl.ActualSpeed
T #MotorControl.SetSpeed
...
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.16



Conhecimento em Automação  
Training Center

### Vista Geral

O tipo de dado STRUCT (estrutura) representa um número fixo de componentes, estes podem ter diferentes tipos de dados em cada caso. Uma estrutura pode ter até 8 níveis de aninhamento.

Uma estrutura pode ser declarada na parte de declaração de um bloco lógico, em um DB global ou em um tipo de dado definido pelo usuário (UDT).

As seguintes restrições são aplicáveis em tipos de dados de uma estrutura:

- elementares (sem restrições)
- complexos (DATE\_AND\_TIME, ARRAY, STRUCT, UDT)
- sem tipo de parâmetros
- sem FBs (modelo multi-instance)

### Acesso aos Componentes

Instruções STL podem ser usados para acessar componentes (tipos de dados elementares) de uma estrutura. Um componente de estrutura é endereçado usando:

- *StructureName.ComponentName*

Um ponto deve ser inserido entre *StructureName* e *ComponentName* como um separador.

Se o tamanho do aninhamento da estrutura é maior, isto é, componentes da estrutura são por vezes estruturas, então o acesso aos menores componentes da estrutura é possível usando o "name path", como:

- *StructureName.ComponentName.SubcomponentName. ...*

Um ponto deve ser inserido entre os nomes dos componentes e subcomponentes em cada caso.

## Declaração dos STRUCTs

### Exemplo: Declaração de um Array - Structure - Array

**DB6 "Declaration View"**

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	Axis	ARRAY[1..4]		
+0.0		STRUCT		
+0.0	Start	BOOL	FALSE	
+0.1	Stop	BOOL	TRUE	
+2.0	Position	ARRAY[1..10]		
+0.0		STRUCT		
+0.0	Cutoffpoint_front	REAL	0.00	
+4.0	Cutoffpoint_back	REAL	0.00	
+8.0	Stoppingpoint	REAL	0.00	
=12.0		END_STRUCT		
=122.0		END_STRUCT		
=488.0		END_STRUCT		

**DB6 "Data View"**

Address	Name	Type	Initial Value	Actual Value
0.0	Axis[1].Start	BOOL	FALSE	FALSE
0.1	Axis[1].Stop	BOOL	TRUE	TRUE
2.0	Axis[1].Position[1].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
6.0	Axis[1].Position[1].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
10.0	Axis[1].Position[1].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
14.0	Axis[1].Position[2].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
18.0	Axis[1].Position[2].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
22.0	Axis[1].Position[2].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
26.0	Axis[1].Position[3].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
30.0	Axis[1].Position[3].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
34.0	Axis[1].Position[3].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
38.0	Axis[1].Position[4].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
42.0	Axis[1].Position[4].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
46.0	Axis[1].Position[4].Stoppingpoint	REAL	0.000000e+000	0.000000e+000
50.0	Axis[1].Position[5].Cutoffpoint_front	REAL	0.000000e+000	0.000000e+000
54.0	Axis[1].Position[5].Cutoffpoint_back	REAL	0.000000e+000	0.000000e+000
58.0	Axis[1].Position[5].Stoppingpoint	REAL	0.000000e+000	0.000000e+000

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.17



Conhecimento em Automação  
Training Center

#### Vista Geral

No exemplo acima, um ARRAY [1..4] com dimensão única com componentes do tipo STRUCT é declarada dentro do DB6 ("Hall\_1") com o Editor de DBs.

A estrutura por vezes consiste de três componentes dos quais os dois primeiros, isto é, "START" e "STOP" tem o tipo de dado BOOL. O terceiro componente tem o tipo de dado ARRAY[1..10].

Os componentes deste tipo de dado ARRAY[1..10] são por vezes do tipo STRUCT com os componentes REAL "Cutoffpoint\_front", "Cutoffpoint\_back" e "Stoppingpoint".

#### Acesso

Os componentes individuais podem ser endereçados como abaixo, por exemplo:

- L "Hall\_1".Axis[3].Position[7].Cutoffpoint\_back
- S "Hall".Axis[2].START, etc.

#### Inicialização de STRUCTs

Os componentes individuais da estrutura podem ser pré atribuídos valores na declaração (coluna "Initial Value"). Os seguintes parâmetros ou variáveis não podem ser inicializados:

- parâmetros input, output e in/out dos FCs
- parâmetros in/out nos FBs
- dados locais nos OBs, FBs e FCs

A inicialização dos valores dos tipos de dados devem ser compatíveis com os tipos de dados.

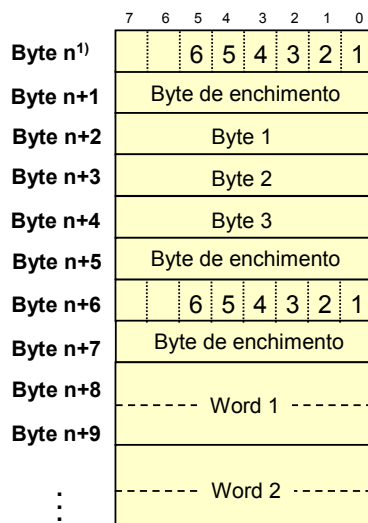
#### Aceitação da Inicialização de Valores

Se novos valores de inicialização são introduzidos na "declaration view" de DBs, estas mudanças somente serão efetivadas (validação como "actual values") depois de ser executada a função do menu *Edit -> Initialize Data Block*.

A inicialização de valores do STRUCTs na declaração dos parâmetros de entrada ou saída nos FBs são aceitos como valores atuais em um DB instance quando estes são gerados.

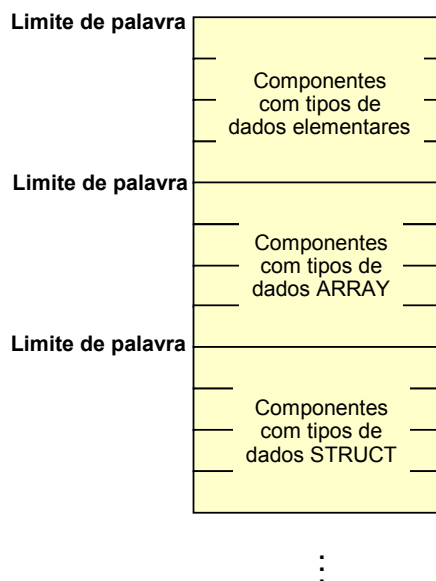
## Armazenagem das Variáveis STRUCT na Memória

### Estrutura com tipo de dados elementares



<sup>1)</sup> n = par

### Estrutura com tipo de dados complexos



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.18



Conhecimento em Automação  
Training Center

### Vista Geral

Conhecimento exato do armazenamento das variáveis STRUCT na memória é então necessária quando, durante a execução do programa, componentes individuais são acessados usando memória ou registrador de endereçamento indireto.

### Armazenamento de Variáveis

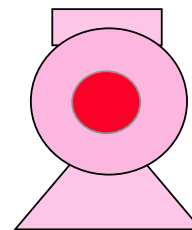
Uma variável STRUCT sempre começa no limite de uma palavra, ou seja, em um byte com um endereço par. Subsequentemente, componentes individuais são então locados em seqüência de sua declaração na memória. Uma variável STRUCT ocupa a memória até o próximo limite de palavra.

Componentes com tipo de dado BOOL começa no último endereço bit significativo, componentes com tipo de dado BYTE e CHAR em um endereço byte par. Componentes com outros tipos de dados sempre começam em um limite de palavra.

## Tipos de Dados Definido pelo Usuário: UDTs

### UDT tipos de dados definidos pelo usuário:

- cria um "template" para posterior uso em declarações
- globalmente válido para todos os blocos da pasta de programa



### Exemplo:

- Definição de um novo tipo de dado (Estrutura):

```
UDT1 STRUCT
    SetSpeed    : REAL;
    ActualSpeed : REAL;
    Enable      : BOOL;
    Disturbance : BOOL;
END_STRUCT;
```

- Declaração das variáveis:

```
    Motor_1: UDT1;
    Motor_2: UDT1;
```

- Acesso a variáveis:

```
    L #Motor_1.ActualSpeed
```

...

UDT1: STRUCT

Set_Speed:	REAL
Actual_Speed:	REAL
Enable:	BOOL
Disturbance:	BOOL

END\_STRUCT

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.19



Conhecimento em Automação  
Training Center

### Vista Geral

Quando uma estrutura de dados se repete freqüentemente em um programa do usuário ou quando uma estrutura de dados está sendo dado seu próprio nome, então STEP 7 permite ao usuário definir tipos de dados (UDT = User Defined Data Type – tipo de dados definidos pelo usuário) próprios (como *typedef* em linguagem de alto nível "C").

Através de tipos de dados relacionados à aplicação, uma tarefa a ser resolvida pode ser programada mais eficientemente. Usuários como fabricantes de máquinas podem então criar tipos de dados específicos para seus projetos.

### Criação de UDTs

UDTs são criadas com o Editor de DB ou com o Editor de Textos e então armazenar na pasta de blocos como um bloco (UDT1 ... UDT65535).

Um nome simbólico pode então ser atribuído para este UDT ou estrutura de dados relacionados na tabela de símbolos globais. Um "template" validado globalmente é criado através de um UDT, o qual pode então ser usado tão freqüentemente como desejado na declaração de novas variáveis ou para criação de DBs globais.

## Uso dos UDTs

The screenshot displays two windows from the SIMATIC Manager software. The left window, titled 'UDT5', shows the definition of a new data type. It is structured as follows:

Address	Name	Type	Initial Value	Comment
0.0		STRUCT		
+0.0	SetSpeed	INT		
+2.0	ActualSpeed	INT		
+4.0	Enable	BOOL		
+4.1	Disturbance	BOOL		
=6.0		END_STR		

The right window, titled 'FC23', shows the declaration of parameters for a function block. It includes:

Address	in	out	Speed	INT	
0.0	in		Speed	INT	
2.0	in_out		Drive	ARRAY[1..10]	
*6.0	in_out			UDT5	

Below the parameter table, the code for the function block is visible, showing the use of the UDT5 structure for the 'Drive' array:

```

A #Drive[1].Enable
A #Drive[2].Enable
A #Drive[3].Enable
A #Drive[4].Enable
A #Drive[5].Enable
A #Drive[6].Enable
  
```

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.20



Conhecimento em Automação  
Training Center

### Vista Geral

No exemplo acima, o UDT5 é criado de 4 componentes (SetSpeed, ActualSpeed, Enable, Disturbance) para uma estrutura de acionamentos (drives) e então inserido em um FC23 na declaração de parâmetros in/out. Um ARRAY unidimensional com 10 componentes do tipo de dado UDT5 é declarado no FC23.

### Valores Iniciais para UDTs

Tipos de dados definidos pelo usuário são pré atribuídos e então usados no programa do usuário exatamente como estruturas. A estrutura de um UDT é a mesma que a de um STRUCT. A declaração das variáveis, que podem ser processadas pelo programa do usuário, não tem ainda tomado lugar com a criação de um UDT. O UDT é um "template", que você pode usar tão frequentemente quanto queira para declaração de novas variáveis.

Do mesmo modo que com uma estrutura, você também tem a possibilidade de estabelecimento de valores iniciais nos UDTs. Se o UDT é então usado para a declaração de uma variável, o conteúdo destas variáveis são inicializados com valores iniciais do UDT (não para parâmetros em FCs, para parâmetros in/out dos FBs e variáveis temporárias).

### Criação de DBs

Um UDT pode também ser usado como um modelo para criação ( Diálogo: *New Data Block* ) de um DB global. Neste caso, um DB é criado com a mesma estrutura e com os valores iniciais do respectivo UDT.



## Tipo de Dado: DATE\_AND\_TIME

### Estrutura:

Byte n <sup>1)</sup>	<b>Ano</b> (90 ... 89)	<b>Mês</b> (01 ... 12)	Byte n+1
Byte n+2	<b>Dia</b> (01 ... 31)	<b>Hora</b> (00 ... 23)	Byte n+3
Byte n+4	<b>Minuto</b> (00 ... 59)	<b>Segundo</b> (00 ... 59)	Byte n+5
Byte n+6	<b>Milissegundos</b> (000 ... 999)	<b>Dia da semana</b> (1..7)	Byte n+7

1=Domingo  
 2=Segunda-feira  
 3=Terça-feira  
 4=Quarta-feira  
 5=Quinta-feira  
 6=Sexta-feira  
 7=Sábado

- Todos os valores são salvos no formato BCD
- Formação da variável:  
**DT#Ano-Mês-Dia-Hora:Minutos:Segundos.[Milissegundos]**  
**Exemplo: DT#1998-03-21-17:23:00:00**
- Processamento através de funções na biblioteca IEC

<sup>1)</sup> n = par

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.21



Conhecimento em Automação  
Training Center

### Vista Geral

O tipo de dado DATE\_AND\_TIME representa um instante, consistindo de data e horário do dia (time-of-day). A abreviação DT pode também ser usada em vez de DATE\_AND\_TIME.

DATE\_AND\_TIME ou DT são palavras chaves e podem portanto também serem escritas em baixo nível.

### Pré ajuste

Uma variável pode ser presetada com um valor inicial na declaração (não como parâmetro de bloco em um FC, como parâmetro in/out em um FB ou como variável temporária).

O formato deve ser do tipo:

- DT#Ano-Mês-Dia-Horas:Minutos:Segundos.Milissegundos

Especificação de milissegundos pode ser suprimida.

### Processamento

Variáveis do tipo de dado DATE\_AND\_TIME podem ser processadas cada uma com a ajuda de acesso absoluto aos componentes individuais ou as funções da biblioteca IEC.

### Nota

O corrente "horário do dia" (time-of-day) do relógio de tempo real da CPU pode ser lido com o SFC1 (READ\_CLK). O horário é retornado pelo SFC1 como um parâmetro de saída do tipo DATE\_AND\_TIME.

## Funções para Processamento de Variáveis DT

### Biblioteca IEC-Library nas Bibliotecas Standard V3.x

- **FC1 (AD\_DT\_TM):** A função FC 1 soma um período de tempo (formato TIME) a um instante (formato DT) e retorna um novo instante (formato DT) como um resultado.
- **FC34 (SB\_DT\_DT):** A função FC 34 subtrai dois instantes (formato DT) e retorna uma duração de tempo (formato TIME) como um resultado.
- **FC35 (SB\_DT\_TM):** A função FC 35 subtrai um período de tempo (formato TIME) de um instante (formato DT) e retorna um novo instante (formato DT) como resultado.
- **FC3 (D\_TOD\_DT):** A função FC 3 combina os formatos de dados DATE e TIME\_OF\_DAY (TOD) e converte estes formats em um formato DATE\_AND\_TIME (DT).
- **FC6 (DT\_DATE):** A função FC 6 extrai o formato de dados DATE do formato DATE\_AND\_TIME.
- **FC7 (DT\_DAY):** A função FC 7 extrai o dia da semana do formato DATE\_AND\_TIME.
- **FC8 (DT\_TOD):** A função FC 8 extrai o formato de dado TIME\_OF\_DAY do formato DATE\_AND\_TIME.
- **Funções de comparação para variáveis DT#Variables:** FC9 (EQ\_DT), FC12 (GE\_DT), FC14 (GT\_DT), FC18 (LE\_DT), FC23 (LT\_DT), FC28 (NE\_DT)

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.22Conhecimento em Automação  
Training Center

### Vista Geral

Com a instalação do STEP7, a biblioteca Standard Library V3.x com a sub biblioteca de Conversão de Blocos IEC também é instalada, a qual contém funções para processamento de tipos de dados IEC.

Funções para processamento de variáveis do tipo DATE\_AND\_TIME também estão nesta biblioteca.

### Notas

#### FC1, FC35

Quando usando FC1, FC3 e FC34 os seguintes pontos devem ser observados:

O instante (Parâmetro T) deve estar dentro da faixa DT#1990-01-01-00:00:00.000 e DT#2089-12-31-23:59:59.999. A função não verifica o parâmetro de entrada.

Se o resultado da adição ou subtração não estiver dentro da faixa especificada acima, o resultado é limitado ao respectivo valor e o resultado binário BR é fixado em "0".

#### FC34

Os instantes devem estar dentro da faixa DT#1990-01-01-00:00:00.000 e DT#2089-12-31-23:59:59.999. A função não verifica o valor inserido. Se o primeiro instante (Parâmetro T1) é maior (mais novo ou recente) do que o segundo (Parâmetro T2), o resultado é positivo. Se o primeiro instante é menor (mais velho) do que o segundo, o resultado é negativo.

Se o resultado da subtração estiver fora da faixa de números TIME, o resultado é limitado ao respectivo valor e o resultado binário é fixado em "0".

#### FC3, FC6, FC7, FC8

Estes valores de funções não reportam qualquer tipo de erro. O usuário se responsabiliza pela correta inserção de valores válidos na entrada.

#### Funções de Comparação

As funções de comparação também não realizam qualquer avaliação de Os respectivos sinais de comparação da função na saída RET\_VAL indica se a comparação foi satisfeita (RET\_VAL=TRUE) ou não (RET\_VAL=FALSE).

## Tipo de Dado: STRING

### Variáveis do tipo STRING (character string):

- **Tipo de dado STRING** representa um character string com até 254 caracteres
- **Aplicação:** Manipulação de mensagens de texto
- **Declaração:**
  - *StringName*: **STRING**[*maxNo*]: '*Initializationtext*'  
(variável String para até *maxNo* caracteres, *maxNo*: 0... 254)
  - *StringName*: **STRING**: '*Initializationtext*'  
(variável String para até 254 caracteres)

### Exemplos:

- **Declaração de variáveis:**
  - Fault signal : **STRING** 'Motor failure\_4'  
(variável *Fault signal* inicializada com o texto acima)
  - Warning : **STRING**[50] ''  
("empty" variável *Warning*, pode aceitar até 50 caracteres)
- **Processamento:**
  - Acessos elementares:  
L #Fault signal[5] (carrega o quinto caracter do *Fault signal*)
  - Processamento por meio dos FCs da biblioteca IEC

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.23Conhecimento em Automação  
Training Center

### Vista Geral

O tipo de dado String é usado para armazenamento de caracteres strings (p.ex.: mensagens de texto). Neste caminho, uma simples "(mensagem) sistema de processamento de palavras" pode ser implementado em uma CPU S7. O tipo de dado STRING representa um character string com até 254 caracteres.

O número é especificado entre colchetes na declaração(1..254) dando o número máximo de caracteres que podem ser salvos na variável STRING. Se esta informação é violada, o Editor STL/LAD/FBD assume um comprimento de 254 caracteres.

### Acesso as Variáveis STRING

Os caracteres individuais de uma variável STRING pode ser acessada com a ajuda de instruções elementares STL, como:

- L *StringName*[5] // carrega o quinto caracter que está armazenado na // variável

O processamento das variáveis STRING atuais (mensagem de texto) é possível usando FCs da biblioteca IEC.

### Inicialização

Na declaração, as variáveis do tipo de dado STRING podem ser pré atribuídas com texto de partida (não como parâmetros de FCs, como parâmetros de um FB ou como variáveis temporárias). A inicialização é feita com caracteres ASCII. Se caracteres especiais para controle são incluídos, então o caracter dolar (\$) deve ser colocado em frente.

Caracteres especiais utilizáveis são:

- \$\$ caracter dolar simples
- \$L, \$l line feed (LF) (pula linha)
- \$P, \$p page feed (pula página)
- \$R, \$r carriage return (término de linha)
- \$T, \$t tabulator (tabulador)

## Armazenagem das Variáveis STRING na Memória

### Exemplo:

- **Declaration with initialization**

- Nome dado: STRING[8]: 'OTTO'

- **Variável STRING armazenada "Nome dado"**

Byte n <sup>1)</sup>	máx. comprimento= 8	→	especifica o número máx. De caracteres salvos, ou seja, a dimensão especificada na declaração
Byte n+1	comprim. corrente= 4	→	especifica o caracter corrente salvo na variável STRING
Byte n+2	1º caracter = 'O'		
Byte n+3	2º caracter = 'T'		
Byte n+4	3º caracter = 'T'		
Byte n+5	4º caracter = 'O'		
Byte n+6	B#16#00		
Byte n+7	B#16#00		
Byte n+8	B#16#00		
Byte n+9	B#16#00		
	⋮		

- A informação sobre o máximo número de caracteres salvos ou sobre o tamanho corrente é avaliado por funções na biblioteca IEC.

<sup>1)</sup> n = par

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.24



Conhecimento em Automação  
Training Center

### Vista Geral

Uma variável do tipo de dado STRING tem no máximo 256 bytes de comprimento, pelo qual até 254 são "net data", isto é, caracteres que podem ser aceitos.

### Armazenamento de Variáveis

Variáveis STRING sempre começam em uma palavra limite, isto é, em um byte com endereço par.

Na determinação de variáveis, seu máximo comprimento é inserido no primeiro byte da variável de acordo com a declaração de variáveis. Da mesma forma, na pré atribuição ou no processamento, o comprimento correntemente utilizado, que é o comprimento dos caracteres string salvos é inserido no segundo byte com a ajuda das funções da biblioteca IEC.

Ambas informações são necessárias para as funções da biblioteca IEC no processamento de variáveis STRING.

Subseqüentemente, os caracteres que se seguem estão no formato ASCII. Os caracteres não preenchidos na variável STRING são preenchidos com W#16#00 na inicialização.

### Passagem de Parâmetros

Variáveis do tipo de dado STRING podem ser passadas da mesma forma que variáveis ARRAY ou STRUCT para os parâmetros dos blocos com o mesmo tipo de dado, isto é, o mesmo comprimento STRING.

Uma passagem de parâmetros para FC ou FB com tipo POINTER ou ANY também é possível.

## Funções para Processamento de Variáveis STRING

### Biblioteca IEC na biblioteca Standard Library V3.x

- **FC2 (CONCAT):** A função FC2 combina duas variáveis STRING no caracter string.
- **FC4 (DELETE):** A função FC 4 delete caracteres L como o enésimo caracter no caracter string.
- **FC11 (FIND):** A função FC 11 entrega a posição do segundo caracter string dentro do primeiro caracter string.
- **FC17 (INSERT):** A função FC 17 insere o caracter string do parâmetro IN2 dentro do caracter string do parâmetro IN1 após o enésimo caracter.
- **FC20 (LEFT):** A função FC 20 entrega o primeiro caracter L de um caracter string.
- **FC21 (LEN):** A função FC 21 calcula o comprimento do caracter string (número de caracteres válidos).
- **FC26 (MID):** A função FC 26 entrega a seção média do caracter string
- **FC31 (REPLACE):** A função FC 31 troca caracteres L do primeiro caracter string (IN1) como o enésimo caracter (incluso) com o segundo caracter string (IN2).
- **FC32 (RIGHT):** A função FC 32 entrega o último caracter L de um caracter string.
- **Funções de comparação de variáveis STRING:** FC10 (EQ\_STRING), FC13 (GE\_STRING), FC15 (GT\_STRING), FC19 (LE\_STRING), FC24 (LT\_STRING), FC29 (NE\_STRING)

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.25



Conhecimento em Automação  
Training Center

#### Vista Geral

Com a instalação do STEP7, a biblioteca Standard Library com a sub biblioteca IEC Blocos de Conversão também é instalada, a qual contém funções para o processamento de tipos de dados IEC.

#### Notas

As funções, em geral, realizam avaliação de erro com a ajuda de detalhes sobre o máximo comprimento ou comprimento atual utilizado. Se as funções reconhecem um erro, então, geralmente, o bit BR é fixado em "0". Uma descrição detalhada das funções individuais podem ser encontradas na ajuda Online da biblioteca IEC.

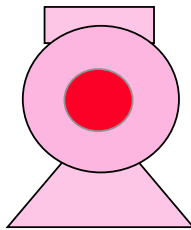
#### Comparação Functions

As funções de comparação realizam comparações léxico gráficas do caractere string. Os caracteres são, começando da esquerda, comparados com seus códigos ASCII (p.ex.: 'a' é maior que 'A' e 'A' é menor que 'B').

O primeiro caractere diferente determina o resultado da operação. Se a porção esquerda de um caractere string longo é idêntico ao caractere string mais curto, então o caractere string é considerado maior.

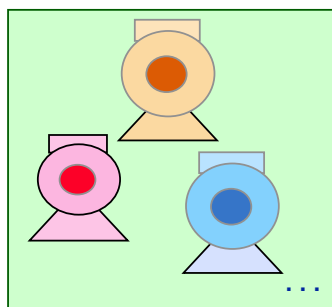
As funções não sinalizam qualquer erro. O respectivo sinal da função de comparação no retorno de valor RET\_VAL informa (RET\_VAL=TRUE) se a comparação acusa igualdade ou diferença (RET\_VAL=FALSE).

## Exercício 5.1: Uso dos Tipos de Dados Complexos



UDT99 "Motor"

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	SetSpeed	REAL	0.000000e+000
+4.0	ActualSpeed	REAL	0.000000e+000
+8.0	SetActDiffMax	REAL	5.000000e-002
+12.0	Enable	BOOL	FALSE
+12.1	Disturbance	BOOL	TRUE
=14.0		END_STRUCT	



Hall\_1

DB51 "Conv\_area\_Motors"

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	ConvArea_1_Motor	ARRAY[1..20]	
*14.0		UDT99	
+280.0	ConvArea_2_Motor	ARRAY[1..20]	
*14.0		UDT99	
=560.0		END_STRUCT	

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.26Conhecimento em Automação  
Training Center

### Objetivo:

Tornar-se familiarizado com os tipos de dados complexos STRUCT, bem como com manipulação de UDTs.

### Tarefa

Em transportadores de área em unidade de moagem, 20 motores do mesmo tipo são ativados. O gerenciamento dos dados do motor mostrado toma lugar para cada caso em um ARRAY por área de transportadores.

De outro modo, a estrutura de dados de cada motor individualmente são idênticos, então eles serão armazenados em um UDT.

O arquivo de dados do motor consiste das seguintes informações:

- SetSpeed (REAL): Velocidade especificada pela sala de controle
- ActualSpeed (REAL): Velocidade atual medida
- SetActDiffMax (REAL): A máxima porcentagem de desvio entre o valor desejado e o valor real especificado para garantir qualidade
- Enable (BOOL): Sinal de habilitação especificado pela sala de controle
- Disturbance (BOOL): Sinal OK retornado para sala de controle

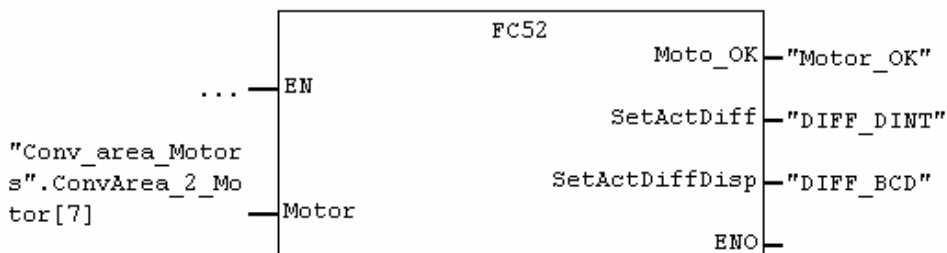
### O que fazer

1. Antes de mais nada, criar um UDT99 "Motor" com a estrutura desejada.
2. Inicializar os valores inseridos no UDT99 como abaixo:
  - SetSpeed: 0.0
  - ActualSpeed: 0.0
  - SetActDiffMax: 0.05 (significa máx. 5% de desvio)
  - Enable: FALSE
  - Disturbance: TRUE
3. Então criar um "Conv\_Area\_Motors". Dentro do DB51 declarar duas variáveis *ConvArea\_1\_Motor* e *ConvArea\_2\_Motor* do tipo ARRAY[1..20] e tipos de componentes UDT99.
4. Checar o conteúdo do DB51 no "data view".

## Exercício 5.2: Acessando Tipos de Dados Complexos

OB1 : Title:

**Network 1**: Title:



### Symbol Information:

P#DB51.DBX364.0	"Conv_area_Motors".ConvArea_2_Motor[7]
Q8.0	Motor_OK
MD0	DIFF_DINT
QD10	DIFF_BCD

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.27



Conhecimento em Automação  
Training Center

### Objetivo:

Tornar-se familiarizado com acesso a parâmetros e variáveis de tipos de dados complexos, bem como com sua declaração usando UDT.

### Tarefa

O modo de operação dos motores individuais nas áreas operacionais estão sendo monitorados com a função FC52.

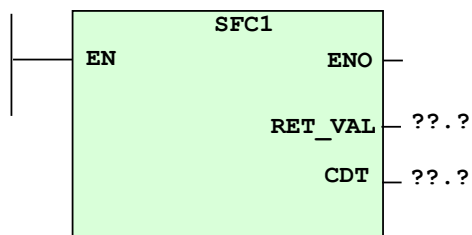
O FC52 tem as seguintes propriedades:

- FC52 conta com arquivo de dados de qualquer motor no parâmetro de entrada *#Motor* (UDT99).
- FC52 entrega, no parâmetro de saída *#Motor\_OK* (BOOL), o valor TRUE, se o bit *Disturbance* não for setado e a porcentagem do desvio entre *SetSpeed* e *ActualSpeed* for menor que *SetActDiffMax* do arquivo de dados que está sendo passado.
- Adicionalmente, FC52 faz avaliação da diferença entre *SetSpeed* e *ActualSpeed* como um número DINT ou como um número codificado BCD no parâmetro de saída *#SetActDiff* (DINT) e *SetActDiffDisp* (DWORD).
- No caso de ser ultrapassada faixa para cima ou para baixo na conversão de DINT ou DWORD para número REAL passado, FC52 seta o bit BR para "0".

### O que fazer

1. Criar um FC52 com as propriedades desejadas.
2. Chamar FC52 no OB1. Alimentar o parâmetro de entrada *#Motor* com o arquivo de dados do sétimo motor da *ConvArea\_2*.  
Ler o sinal de saída *#Motor\_OK* na saída Q8.0.  
Também alimentar o parâmetro de saída *#SetActDiff* e *#SetActDiffDisp* com o parâmetro atual MD0 e QD10 (display digital).
3. Transferir os blocos para a CPU.
4. Testar FC52 com a ajuda da função "Monitor/Modify Variable", especificando diferentes valores para os respectivos arquivos de dados.

## Exercício Adicional 5.3: Lendo “Time-of-Day” com SFC 1 (READ\_CLK)



### Parameter

Parameter	Declaration	Data Type	Memory Area	Description
CDT	OUTPUT	DATE_AND_TIME (DT)	D, L	Output of the current time-of-day and the current date
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Return value of SFC

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_05P.28Conhecimento em Automação  
Training Center

**Objetivo:** Tornar-se familiarizado com tipos de dados complexos DATE\_AND\_TIME.

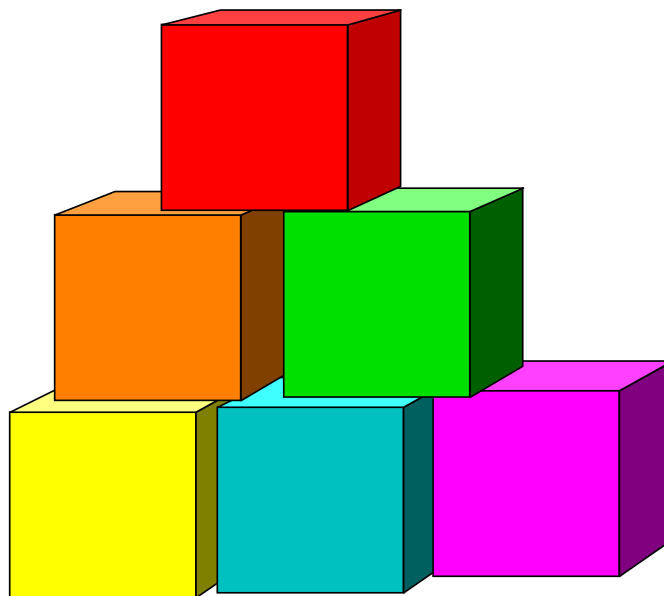
**Tarefa** Criar um FC53 com a seguinte funcionalidade:

- O FC53 lê o horário do dia (time-of-day) da CPU com a ajuda da função do sistema SFC51.
- As horas e minutos são mostradas no display digital.

- O que fazer**
1. Criar um bloco FC53 com a funcionalidade acima.
  2. Chamar o FC53 no OB1.
  3. No SIMATIC Manager, checar, com ajuda da opção do menu *PLC -> Set Time and Date*, se o relógio da CPU está corretamente ajustado.
  4. Transferir o FC53 e o OB1 para a CPU.
  5. Testar o programa.



## Chamada de Blocos e Modelo Multi-instance



SIMATIC S7

Siemens AG 1999. All rights reserved.

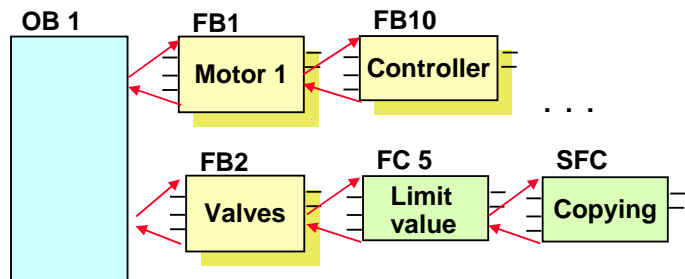
Date: 04.10.2007  
File: PRO2\_06P.1Conhecimento em Automação  
Training Center

Conteúdo	Pág.
Blocos para Programação Estruturada .....	2
Vista Geral dos Blocos em STEP 7 .....	3
Propriedades das Funções .....	4
Mecanismo de Passagem para Tipos de Dados Elementares .....	5
Chamada de Funções com Tipos de Dados Complexos .....	6
Características para chamada de Funções .....	7
Propriedade dos Blocos de Funções .....	8
Formação Instance dos Blocos de Funções .....	9
Passagem de Parâmetros na chamada de um FB .....	10
Chamada de FB com Tipos de Dados Complexos .....	11
Características para chamada de Blocos de Funções .....	12
Exercício 6: O Modelo Transportador para Planta de Engarrafamento .....	13
Exercício 6.1a: Planta de Engarrafamento – Modo de Seleção .....	14
Exercício 6.1b: Planta de Engarrafamento – Transportador .....	15
Estrutura do Modelo Multi-instance .....	16
Programação Orientada a Objeto usando Multi-instances .....	17
Implementando uma "prensa linha" no STEP 7 .....	18
Propriedades do Modelo Multi-instance .....	19
Exercício 6.2: O Modelo Transportador como Linha de Montagem .....	20
Exercício 6.2a: Estrutura de Programa para uma Estação de Trabalho .....	21
FB1 "estação" – Método de Funcionamento .....	22
FB2 "transporte" – Método de Funcionamento .....	23
Exercício 6.2b: Expansão para 3 Estações .....	24
Interconexão de Parâmetros de Blocos .....	25

## Blocos para Programação Estruturada

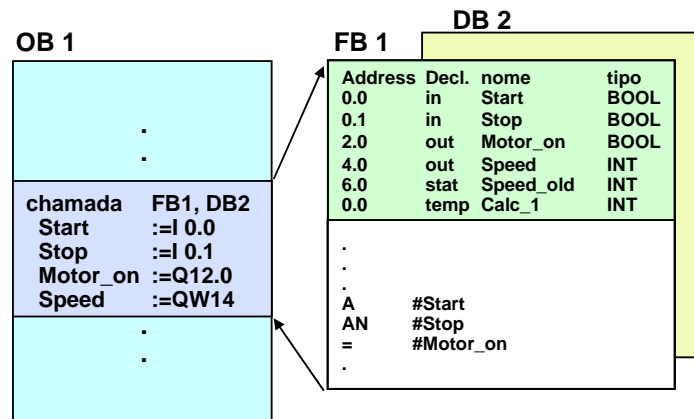
### Modularização da Tarefa Inserida:

- Tarefas parciais são resolvidas em seus próprios blocos
- Atribuição de parâmetros atribui flexibilidade de uso
  - Exemplo: Ciclo de perfuração com tamanho de parâmetro atribuível



### Re-usabilidade dos Blocos:

- Blocos podem ser chamados tão freqüentemente quanto seja resquerido
- Restrições:
  - sem acesso a endereços globais
  - comunicação somente via lista de parâmetros



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.2



Conhecimento em Automação  
Training Center

### O que é um Programa Estruturado?

Quando funções similares ou idênticas se repetem em um processo o qual está sendo controlado, procura-se de um modo geral soluções válidas e escrevê-las em um bloco com parâmetros atribuíveis.

Este bloco é freqüentemente chamada no ciclo de programa. Ele pode ser alimentado com diferentes parâmetros nos arquivos de dados cada vez que este para chamado.

### Vantagens

A divisão do programa do usuário em blocos individuais tem as seguintes vantagens:

- Uma tarefa complexa inserida pode ser subdividida em pequenas e claras tarefas parciais. Os blocos para as tarefas parciais podem ser criados e testados independentemente uns dos outros.
- Com a ajuda dos parâmetros, blocos podem ser projetados para serem flexíveis. Desta forma, por exemplo, um ciclo de perfuração pode ser criado através da passagem de coordenadas e profundidade do buraco a ser perfurado como parâmetros do bloco.
- Blocos podem ser chamados tão freqüentemente quanto necessário com diferentes parâmetros em seus arquivos de dados, ou seja, podem ser reutilizados. A chamada incondicional de um bloco somente é então possível se não existir acesso a endereços, como entradas, saídas, memórias bit ou variáveis em DB de dentro do bloco. Blocos devem comunicar exclusivamente com o "mundo externo" via lista de parâmetros.
- Blocos em STEP 7 podem ser modificados e transferidos para CPU durante a execução do programa independentemente uns dos outros. Deste modo, você pode, por exemplo, atualizar o software de um sistema quando ele está em operação.
- Para muitas tarefas, blocos para tarefas especiais podem ser entregues em bibliotecas padrões pré projetadas as quais podem então se integradas ao respectivo programa do usuário.
- Blocos para uso freqüente em tarefas padrões podem ser integradas no sistema operacional da CPU através do fabricante da CPU na forma de blocos de funções do sistema (SFB) ou funções do sistema (SFC).

## Vista Geral dos Blocos em STEP 7

Tipos de Blocos	Propriedades
Bloco de Organização (OB)	- interface do usuário - graduação de prioridades (0..27) - informações específicas de partida na pilha de dados local
Bloco de Função (FB)	- parâmetros atribuíveis - com memória (=localização de memória)
Função (FC)	- parâmetros atribuíveis (parâm. devem ser atribuídos na chamada) - retorno de valor pode ser retornado - basicamente sem memória
Bloco de Dados (DB)	- armazenagem de dados locais estruturados (DB Instance) - armazenagem dados globais estruturados (validados no programa)
Bloco de Função do Sistema (SFB)	- FB (com memória) armazenado no sistema operacional da CPU e chamável pelo usuário
Função do Sistema (SFC)	- FC (com memória) armazenado no sistema operacional da CPU e chamável pelo usuário
Bloco de Dados do Sistema (SDB)	- bloco de dados para configuração de dados e parâmetros

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.3Conhecimento em Automação  
Training Center

### Vista Geral

STEP 7 oferece uma série de diferentes blocos para subdividir uma tarefa de automação complexa em pequenas tarefas parciais. Estas tarefas parciais refletem a função tecnológica do sistema ou do processo.

### Classes de Blocos no STEP 7

Blocos são, por suas funções, sua estrutura ou sua limitada aplicação partes do programa do usuário. Os blocos no STEP 7 podem – dependendo do seu conteúdo – ser dividido em duas classes:

- Blocos Lógicos:

Blocos Lógicos são blocos de organização (OB), blocos de funções (FB), funções (FC) bem como blocos de funções do sistema (SFB) e funções do sistema (SFC).

As instruções do programa do usuário são armazenadas nestes blocos.

- Blocos de Dados:

Blocos de Dados são blocos de dados do usuário (DB) e os blocos de dados do sistema (SDB).

O usuário podem armazenar dados os quais ocorrem durante a execução do programa em blocos de dados e estes dados podem ser acessados mais tarde.

O conteúdo dos blocos de dados do sistema (SDB) são disponíveis exclusivamente para a CPU (dados de parametrização). SDBs não são criados ou descritos pelo programa do usuário, mas por ferramentas como HW-CONFIG ou NETPRO.

## Propriedades das Funções

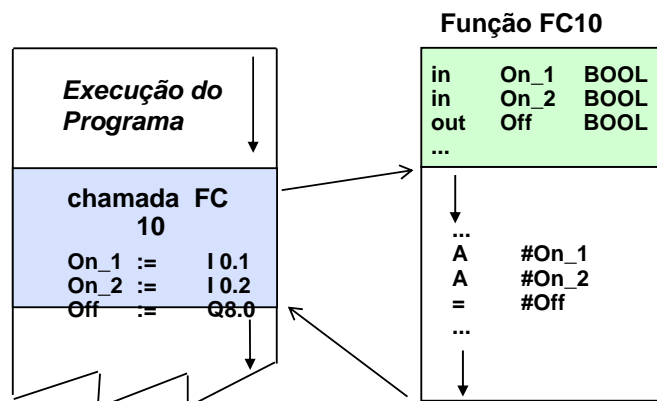
### Parâmetros atribuíveis de blocos:

- muitos parâmetros input, saída e in/out conforme necessidade
- sem memória, isto é somente variáveis temporárias

### Conforme IEC 1131-3:

- muitos parâmetros input conforme necessidade
- somente um parâmetro saída RET\_VAL
- sem acesso a variáveis globais e endereços absolutos
- com os mesmos parâmetros input eles entregam resultados idênticos

Expande o conjunto de instruções do processador



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.4



Conhecimento em Automação  
Training Center

### Vista Geral

Funções (FC) representam parâmetros atribuíveis de blocos sem memória. No STEP 7 eles podem ter muitos parâmetros de entrada (input), parâmetros de saída (saída) e parâmetros de entrada/saída (in/out) conforme necessidade.

Funções não tem memória; não existe área de dado separada, permanente, para armazenamento de resultados. Resultados temporários que ocorrem durante a execução da função podem somente ser armazenadas em variáveis temporárias da respectiva pilha de dados local.

Funções expandem o conjunto das instruções do processador.

### Aplicação

Funções são primariamente usadas quando valores de funções são retornadas da chamada dos blocos (p.ex.: funções matemáticas, controle simples com operações lógicas binárias).

### IEC-1131 Funções Conformes

Se funções estão sendo criadas em conformidade com IEC 1131-3, então as seguintes regras devem ser observadas:

- Funções podem ter tantos parâmetros de entrada (input) quanto necessário, eles podem, desta forma, somente retornar um resultado do parâmetro de saída (saída) RET\_VAL.
- Variáveis globais não podem nem serem lidas ou escritas dentro das funções.
- Endereços absolutos não podem nem serem lidos ou escritos dentro de funções.
- Nem blocos de dados, nem blocos de dados instance podem ser chamados dentro das funções.

Devido a perda de "memória", o resultado retornado da função conforme a norma apenas é dependente dos valores nos parâmetros de entrada. Para valores idênticos do parâmetro de entrada, uma função também retorna o resultado idêntico.

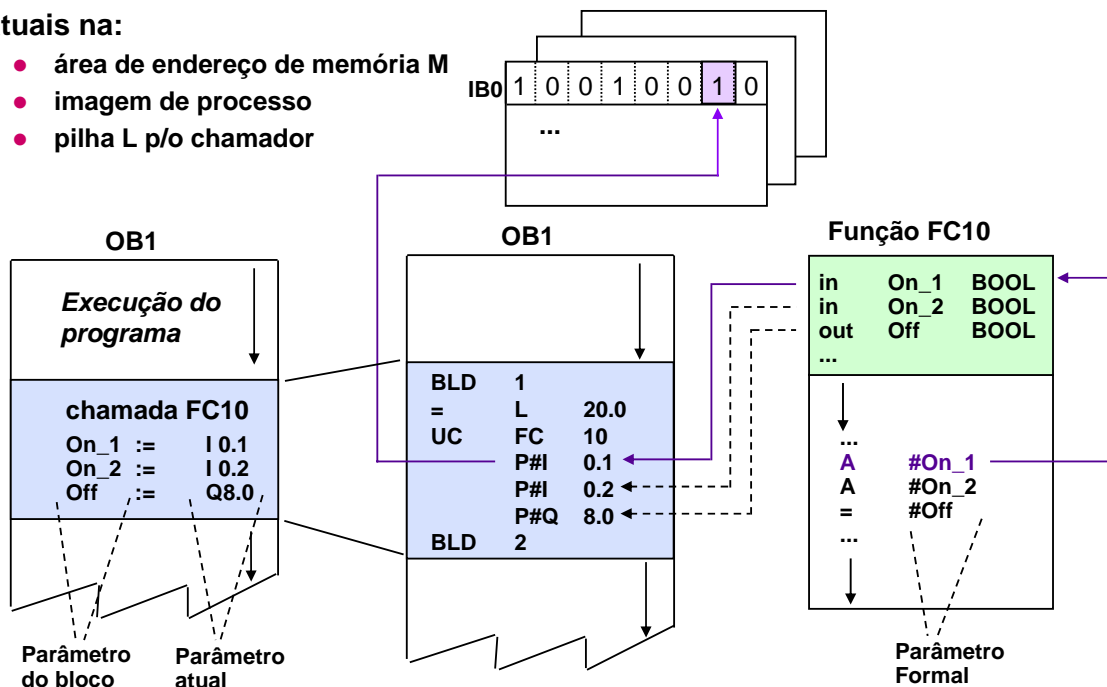
Ele é por esta razão adequado ao programador para criar funções em conformidade com a norma ou fazer blocos de programação e estruturação no STEP 7 como ele é no STEP 5.

## Mecanismo de Passagem para Tipos de Dados Elementares

### Parâmetros Elementares

atuais na:

- área de endereço de memória M
- imagem de processo
- pilha L p/o chamador



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.5



Conhecimento em Automação  
Training Center

### Parâmetros de FC

Dados para processamento podem ser manipulados sobre a chamada da função. Estes dados passando tomam lugar exclusivamente através da lista de parâmetros que aparecem após a chamada (chamada). Os nomes e os tipos de dados dos parâmetros de bloco que aparecem são estabelecidos na parte de declaração do FC.

Parâmetros de entrada (Input; somente leitura), de saída (saída; somente escrita) e entrada/saída (in/out; leitura e escrita) podem ser declarados.

O número de parâmetros não tem restrições (espaço de memória), os nomes podem conter no máximo 24 caracteres. Em adição, os parâmetros podem ser fornecidos com comentário detalhado. Se o bloco não tem qualquer parâmetro, então a lista de parâmetros é listada na chamada do bloco FC.

### Mecanismo de Passagem

Com uma chamada, o Editor STL/LAD/FBD antes de tudo calcula os ponteiros de área cruzada dos parâmetros atuais dados na lista de parâmetros e armazena estes imediatamente após a instrução de chamada do FC.

Se o acesso a um parâmetro formal (p.ex.: A On\_1) agora toma lugar dentro do FC, a CPU determina a instrução de chamada do FC do endereço de retorno armazenado na Pilha B (B-Stack). Da lista de parâmetro pertinentes, o FC então determina o ponteiro da área cruzada para os parâmetros atuais os quais pertencem aos parâmetros formais. Acesso para os parâmetros atuais então tomam lugar através deste ponteiro.

Este mecanismo de passagem corresponde a "chamada by Reference". Se acesso a parâmetro formal toma lugar dentro de um FC, então acesso ao parâmetro atual correspondente resulta disto.

Este mecanismo de passagem via ponteiro ocasiona isto:

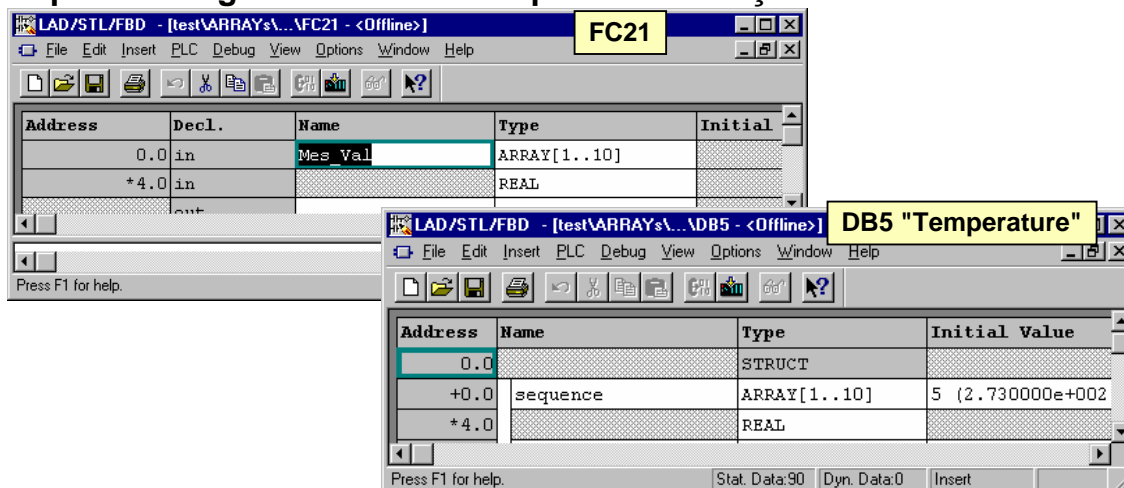
- todos os parâmetros de bloco devem ser atribuídos na chamada de um FC.
- parâmetros de bloco não podem ser inicializados na declaração.

### Notas

Se um parâmetro de bloco é determinado com o parâmetro atual dentro de um DB ou se tipos de dados complexos são passados, então a passagem de parâmetros se tornam mais complexos. (ver apêndice).

## Chamada de Funções com Tipos de Dados Complexos

### Exemplo: Passagem de um ARRAY para uma Função



#### Atribuição dos parâmetros somente é possível simbolicamente

Network 1: Mes\_Val é declarado como um array no FC21

```
chamada FC 21
Mes_Val := "Temperature".sequência
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.6



Conhecimento em Automação  
Training Center

### Vista Geral

Parâmetros do tipo de dado complexo (ARRAY e STRUCT) oferece um claro e eficiente meio de transferir grandes quantidades de dados relacionados entre a chamada e o bloco chamado e assim acomodando o conceito de "Programação Estruturada".

Um array ou uma estrutura pode ser passado para uma função chamada como uma variável completa.

### Atribuição de Parâmetros

Para o passe, um parâmetro do mesmo tipo de dado como o parâmetro atual a ser passado deve ser declarado na função chamada. A atribuição de um parâmetro (tipo de dado: ARRAY, STRUCT, DATE\_AND\_TIME e STRING) pode somente tomar lugar simbolicamente.

Desde que variáveis do tipo de dado complexo somente podem ser fixadas em blocos de dados ou em pilha de dados local, o parâmetro atual deve cada um ser localizado em um DB (global ou instance) ou em pilha de dados local do bloco chamado.

Após o Editor STL/LAD/FBD tem checado a compatibilidade dos tipos de dados do parâmetro atual e parâmetro de bloco na passagem de bloco para um FC, somente um POINTER com um número de DB e um ponteiro de área cruzada para o parâmetro atual é passado para o FC chamado.

Este POINTER é fixado na Pilha L do bloco chamado (V área) através da macro chamada. Este POINTER é então de grande importância para o programador em particular, quando o parâmetro passado tem de ser acessado indiretamente (ver apêndice).

### Notas

- O número do dado local ocupado pode ser determinado pela seleção do menu options View -> bloco Properties.
- Componentes do ARRAYS ou STRUCTs também podem ser passados para um parâmetro de bloco se o parâmetro de bloco e os componentes ARRAY ou STRUCT são do mesmo tipo de dado.

## Características para chamada de Funções

### Instrução chamada

- **Instrução é Macro**
  - Sobrescrever conteúdo dos registradores são possíveis, mesmo registradores DB
  - Atenção com o conteúdo da Pilha B (B Stack)
  - Após a chamada outro DB é aberto
  - Tempo de processamento para chamada depende no número e localização de memória dos parâmetros atuais
- **Instrução chamada assegura que os parâmetros de bloco são corretamente alimentadas com os dados correntes**
- **Exemplo:**
  - chamada FC10
  - On\_1 := I 0.1
  - On\_2 := I 0.2
  - Off := Q8.0

### Instrução chamada UC e CC

- **Chamada de bloco independente do RLO (UC) ou dependente (CC)**
  - Exemplos: UC FC20 ou CC FC20
- **utilizável somente, quando o FC não tem parâmetros**

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.7



Conhecimento em Automação  
Training Center

### Instrução chamada

A instrução (Macro) chamada deve ser usada para chamada de blocos (FCs, SFCs, FBs e SFBs).

Em uma chamada de FC, uma informação direta de troca entre o bloco chamado e a função chamada somente é possível via chamada. A chamada assegura que os parâmetros formais do bloco são corretamente alimentados. Em qualquer caso, diversas características também devem ser levados em consideração, o que resulta do fato de que chamada é implementado através de uma macro que por sua vez se consiste de diversas instruções STL.

Se um parâmetro formal é atribuído com endereços que são encontrados em um DB, então parâmetros passados tomam lugar com a ajuda do registrador DB (ver apêndice). Disto resulta:

- dentro da chamada do FC, é possível que, o DB que é aberto não é o DB que foi aberto antes do chamada.
- se a CPU vai para STOP durante o processamento do FC chamado, então o valor mostrado na Pilha B (B-Stack -> DB-registrador) é que o qual o Editor STL usado para sobrescrever o registrador DB no parâmetro atribuído.
- se após o processamento, um salto é feito para trás dentro do bloco chamado, é possível que o DB não esteja aberto que foi aberto antes do chamada.

### Instrução UC e CC

Blocos também podem ser chamados com a instrução UC ou CC. A instrução de chamada é uma instrução absoluta, isto é, UC sempre chama o bloco independente de condições (p.ex.: UC FC20).

A instrução de chamada CC é uma instrução condicional, isto é, CC somente chama um bloco quando o RLO é igual a "1". Se o RLO é igual a "0", então CC não faz chamada do bloco e seta o RLO em "1". Subseqüentemente, a instrução seguinte a chamada CC é processada.

### Importante

UC e CC somente podem ser usados quando nenhum parâmetro é declarado na chamada do FC.



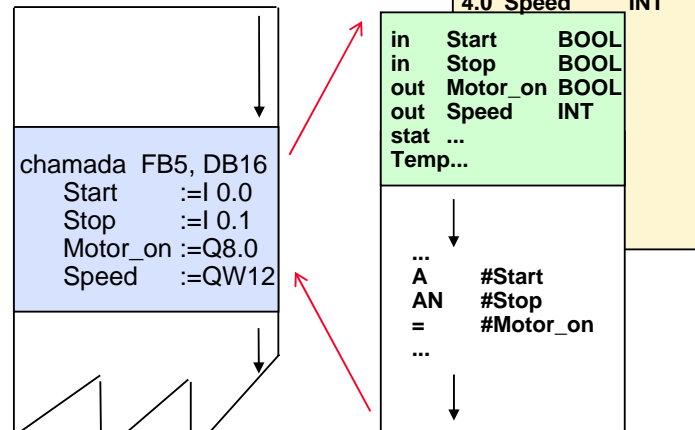
## Propriedades dos Blocos de Funções

### Parâmetros atribuíveis dos blocos:

- Conformidade com IEC 1131-3
- Permite tantos parâmetros input, saída e in/out quanto necessário
- com memória, isto é não somente variáveis temporárias mas também estáticas
- Chamada com sua própria área de dados (instantiating)
- "Encapsulamento de Dados"

### Aplicação:

- Funções de temporização e contagem
- Unidades de Controle de processo com estados internos
  - aquecedores
  - acionamentos, válvulas, etc.



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.8



Conhecimento em Automação  
Training Center

### Vista Geral

Blocos de Funções (FB) são blocos do programa do usuário e representam, de acordo com a IEC Standard 1131-3, blocos lógicos com memória. Eles podem ser chamados por OBs, FBs e FCs.

Blocos de Funções podem ter tantos parâmetros input, saída e in/out quanto é necessário bem como variáveis temporárias e estáticas.

Diferentemente dos FCs, os FBs são inicializados, isto é, um FB é determinado por sua própria área de dados privada, por exemplo, pode "lembrar" estados do processo de chamada para chamada. De forma simples, esta área de dados privados é seu próprio DB, o chamado DB instance.

### "Memória"

O programador tem a oportunidade para declarar variáveis estáticas na seção de declaração do FB. O FB pode "lembrar" informações de chamada para chamada nestas variáveis.

A possibilidade para um FB "lembrar" informações sobre diversas chamadas é a diferença essencial dos FCs.

### Aplicações

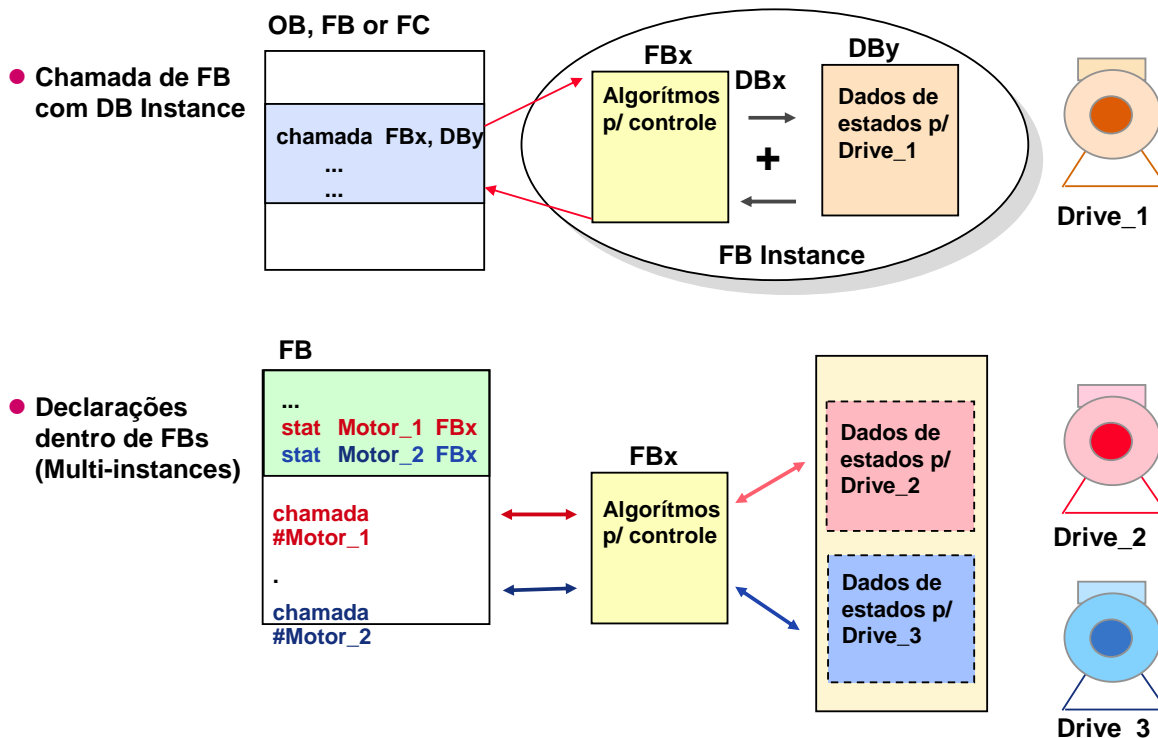
Com a ajuda desta "memória", um FB pode, por exemplo, implementar funções de contagem e temporização ou unidades de controle de processos, tal como estações de processamento, acionamentos, aquecedores, etc.

Em particular, FBs são adequados para controle de todas unidades de processo cuja performance depende não somente de influências externas mas também em estados internos, tal como processamento passo-a-passo, velocidade, temperatura, etc.

Quando controles como unidades, os dados de estado interno das unidades de processo são então copiadas para variáveis estáticas dos FBs.



## Formação Instance dos Blocos de Funções



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.9Conhecimento em Automação  
Training Center

### O que é um Instance?

O conceito de FB com instance tem grande importância e estabelece o critério essencial de distinção com os FCs. A definição das variáveis dentro de linguagens de alto nível como "C" sob declaração de nomes de variáveis e tipos de dados na declaração é chamado de "instance".

A mesma forma de variáveis são também criadas nos FBs. Somente através desta 'própria' área de dados, na qual os valores dos parâmetros do bloco bem como variáveis estáticas são armazenadas, isto torna um FB a ser uma unidade executável (FB instance).

O controle de uma unidade de processo física, tal como um acionamento ou uma caldeira então toma lugar com a ajuda de um FB instance, isto é, um FB com uma área de dados fixada.

### Instance

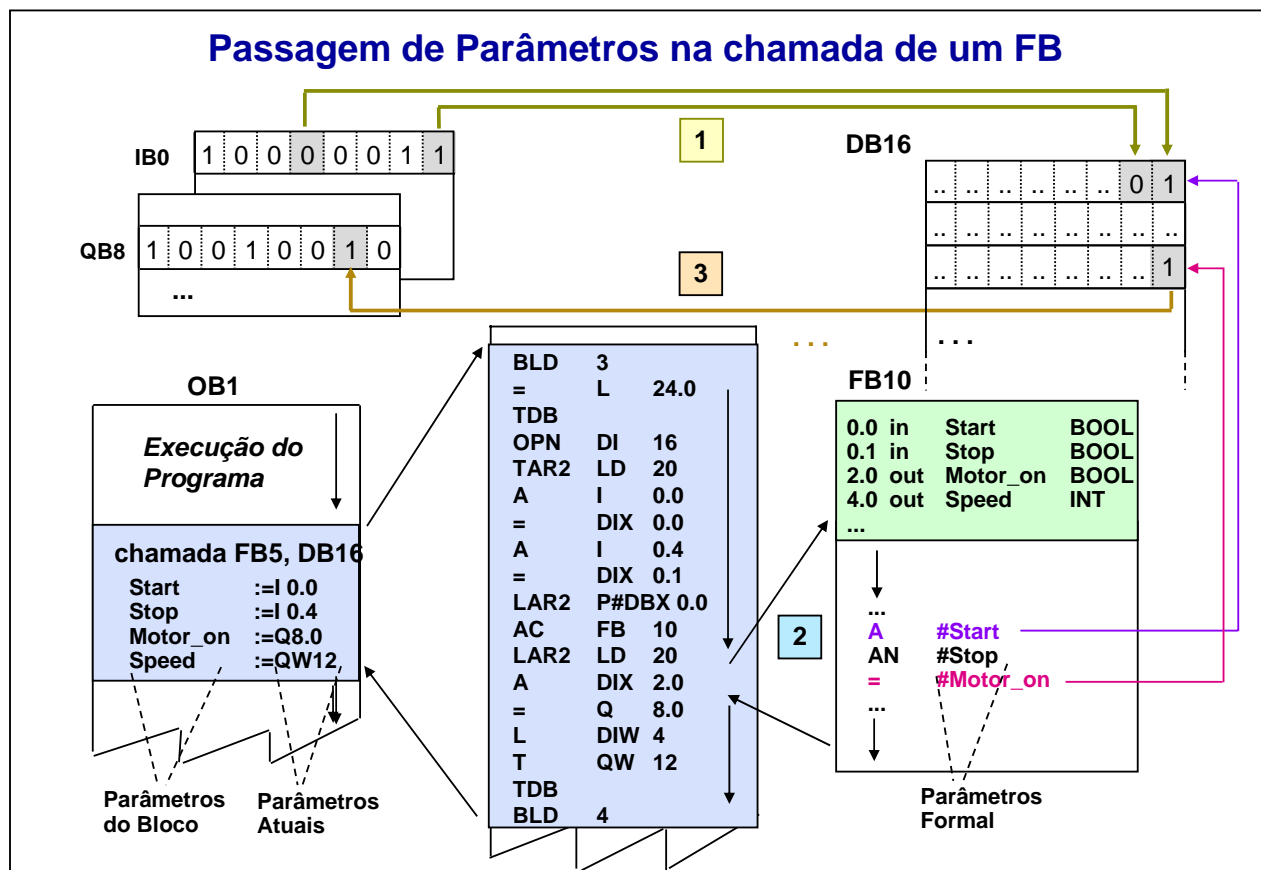
A criação de um FB instance, isto é, a atribuição de sua própria área de memória na chamada de um FB, pode ser feita de dois modos no STEP 7:

- através da declaração explícita de um FB instance quando um FB é chamado.
- através da declaração explícita de instances de um FB dentro um bloco de funções de alto nível (modelo multi-instance).  
STEP 7 então assegura que a área de dados requerida para o instance é definida dentro da área de dados do FB de alto nível.

### Vantagens

O conceito instance do STEP 7 tem as seguintes vantagens:

- Na chamada dos FBs, nenhuma medição para salvamento e administração dos dados locais são necessários exceto para a atribuição dos DBs instance.
- Um FB pode ser usado diversas vezes pelo conceito instance. Se, por exemplo, diversos acionamentos do mesmo tipo estão sendo controlados, então estes tomam lugar usando diversos instances de um FB. O estado dos dados dos acionamentos individuais são armazenados nas variáveis estáticas do FB.



## Chamada de FB com Tipos de Dados Complexos

### Exemplo: Passagem de um ARRAY para um FB

**FB17**

Address	Decl.	Name	Type	Initial Value	Comme
0.0	in	Meas_1	ARRAY[1..10]		
*4.0	in		REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL		
48.0	in_out	Meas_2	ARRAY		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT		

**DB2 "Temperature"**

Address	Name	Type	Initial Value
0.0		STRUCT	
+0.0	Cylinder	ARRAY[1..10]	
*4.0		REAL	
+40.0	Shaft	ARRAY[1..15]	
*4.0		REAL	
=100.0		END_STRUCT	

#### Atribuição de parâmetros complexos somente é possível simbolicamente

Network 1:

```

chamada  FB  17, DB 30
Meas_1   := "Temperature".Cylinder
Sum_1    := MD20
Sum_2    := MD30
Meas_2   := "Temperature".Shaft
  
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.11



Conhecimento em Automação  
Training Center

### Tipo de Dado Complexo

Do mesmo modo que com FCs, endereços de tipos de dados complexos (ARRAY, STRUCT, DATE\_AND\_TIME and STRING) podem ser passados completamente para uma chamada de FB.

Para a passagem, um parâmetro do mesmo tipo de dado como o parâmetro atual ser passado deve ser declarado na chamada do FB.

A atribuição como um parâmetro somente é possível simbolicamente.

### Parâmetros Input e saída

Para parâmetros input e saída dos tipos de dados complexos, correspondem a áreas para os valores dos parâmetros atuais são determinados no DB instance. Na chamada do FB, os parâmetros atuais dos parâmetros de entrada são então copiados dentro do DB instance usando SFC 20 (BLKMOV) ("Passing by Value"), antes da chave atual na seção de instrução do FB.

Da mesma maneira, os valores do parâmetro de saída são copiados de volta do DB instance no parâmetro atual depois que o FB tenha sido processado.

Como um resultado, uma não insignificante quantidade de cópias (processamento de tempo) pode ocorrer nos parâmetros de entrada e saída. Esta quantidade de cópias é transferida com parâmetros de in/out.

### Parâmetros In/Out

Nenhuma "Passing by Value" ocorre com parâmetros in/out dos tipos de dados complexos. 6 bytes são meramente reservadas para cada parâmetro in/out na área de dados instance. Um POINTER para os parâmetros atuais são inseridos nestes bytes ("Passing by Reference").

### Notas

- Parâmetros Input e saída de tipos de dados complexos podem ser inicializados na seção de declaração de um FB, contudo não parâmetros in/out.
- Parâmetros Input e saída de tipos de dados complexos não tem que ser atribuídos na chamada de um FB, parâmetros in/out tem que ser atribuídos.
- A memória ou registrador indireto acessam parâmetros input/saída ou parâmetros in/out de tipos de dados complexos é diferente dos parâmetros elementares.

## Características para chamada de Blocos de Funções

### Passagem de Parâmetros "por valor" (Cópia do Valor):

- **Atribuição de parâmetros do FB em uma instrução chamada:**
  - Parâmetros de FB não tem que ser atribuídos
  - Atribuição e desatribuição pode ser feita do "lado de fora"  
p.ex.: direto do Painel de Operação
  - Exceção: parâmetros in/out dos tipos de dados complexos (STRUCT, ARRAY, STRING e DATE\_AND\_TIME)
- **Inicialização:**
  - Parâmetros FB podem ser inicializados na declaração
  - Exceção: parâmetros in/out dos tipos de dados complexos (STRUCT, ARRAY, STRING e DATE\_AND\_TIME)
- **Acesso aos parâmetros formais tomam lugar internamente usando registradores DI e AR2**
  - Se o registrador DI ou AR2 é sobrescrito, acesso ao dado instance não é mais possível
- **Instrução de chamada adicional UC e CC**
  - Exemplos: UC FB20 ou CC FB20
  - Somente utilizável se o FB não tenha dados instance (parâmetros + variáveis estáticas)

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.12



Conhecimento em Automação  
Training Center

#### Atribuição de parâmetros de bloco

Parâmetros de blocos não tem que ser atribuídos na chamada de um FB. Neste caso, valores nenhuns são copiados dentro ou fora do DB instance. Os parâmetros no DB instance mantém os valores que também estavam salvos na última chamada.

Exceção: Parâmetros in/out dos tipos de dados complexos devem ser atribuído na lista de parâmetros.

#### Parâmetro de acesso do "Lado de fora"

Acesso aos parâmetros dentro de um DB instance pode ser feito do mesmo modo como com os endereços dos DBs globais. Parâmetros de blocos podem também deste modo ser atribuído ou "desatribuído" do "lado de fora".

Isto é então especialmente utilizável quando, por exemplo, somente componentes individuais dos tipos de dados complexos tem que ser atribuído ou "desatribuído" ou parâmetros são diretamente ligados com campos input/saída nos OPs.

Exceção: Parâmetros in/out dos tipos de dados complexos não podem ser atribuídos ou "desatribuídos" do "lado de fora".

#### Inicialização

Parâmetros de blocos e variáveis estáticas podem ser inicializadas no FB declaração. Se um DB instance é então criado, então os valores iniciais especificado na declaração são inseridos no DB instance.

Exceção: Parâmetros in/out dos tipos de dados complexos não podem ser inicializados.

#### Nota

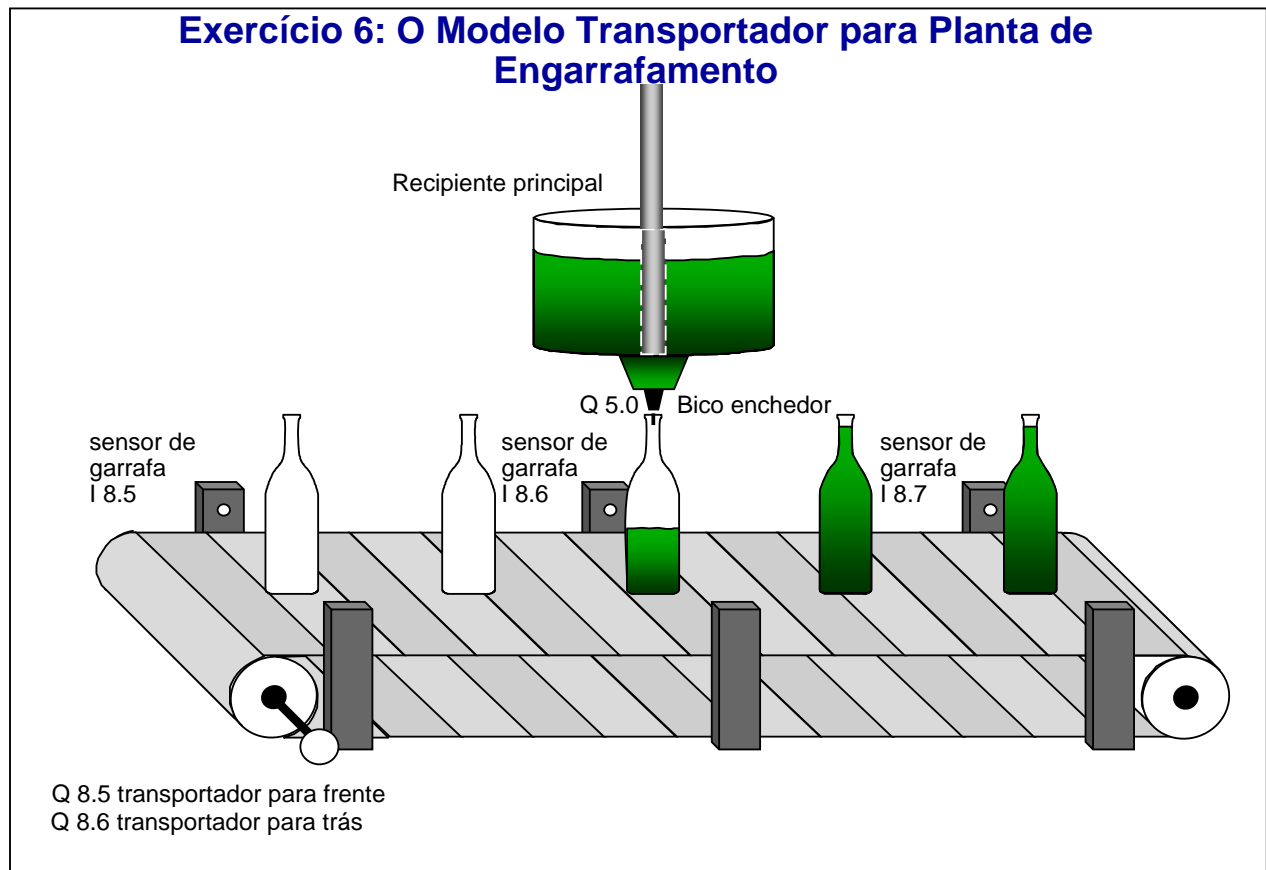
Se o registrador DI ou o registrador AR2 estiver sobrescrito dentro do FB processado, então o acesso aos dados instance (parâmetros input, saída, in/out e variáveis estáticas) podem ser mais longas do que feito dentro do FB.

#### Instruções UC, CC

Blocos podem também ser chamados com instruções independentes do RLO (UC) ou com instruções dependentes do RLO (CC).

UC e CC podem somente ser usados quando o chamado do FB não tem dados instance, isto é, nenhum dos parâmetros de blocos ou nenhuma variável estática foi declarada na seção de declaração.

## Exercício 6: O Modelo Transportador para Planta de Engarrafamento



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.13Conhecimento em Automação  
Training Center

### Tarefa

Uma planta de engarrafamento está sendo automatizada conforme descrito:

A planta pode ser operada através de dois modos de operação "Manual" e "Automático". Adicionalmente, uma lógica correspondente para ligar e desligar o sistema existente.

- Ligando e desligando: A planta pode ser ligada usando a entrada I 0.0 (botão de pulso com função NA). A planta é desligada novamente usando a entrada I 0.1 (botão com contato NF). Quando a planta é ligada, o LED da saída Q 8.1 é acesa.
- Escolhendo o modo de operação: quando a planta é ligada, um modo de operação pode ser escolhido.  
Modo Manual é presetado com I 0.4 = 0 e Modo Automático é presetado com I 0.4 = 1. O modo selecionado é adotado com um pulso na entrada I 0.5. O indicador do modo selecionado é (Manual = Q 8.2, Automático = Q 8.3).  
Quando o modo é alterado ou a planta é desligada, o modo previamente selecionado é desselecionado.
- Controle do transportador em modo manual: Em modo manual, o transportador pode ser movimentado para frente com o botão de pulso I 0.2 (Q 8.5) e para trás com I 0.3 (Q 8.6).
- Controle do transportador em modo automático: no modo automático, o motor de acionamento (Q 8.5) liga e se mantém ligado enquanto o botão desliga não para pressionado (I 0.1) ou até que o sensor (I 8.6) detecte uma garrafa na posição de enchimento.
- Enchendo uma garrafa: quando a garrafa estiver sobre a estação de enchimento (I 8.6=1), o enchimento começa. O procedimento de enchimento é realizado em um período de 3 segundos e é indicado pela saída Q 5.0.
- Contagem de garrafas: Outro sensor é utilizado para arquivar o número de garrafas vazias. O sensor I 8.7 arquivas as garrafas cheias. As garrafas cheias são contadas com a planta em operação e são mostradas no display digital QW 6.

## Exercício 6.1a: Planta de Engarrafamento – Modo de Seleção

Liga / desliga planta

I 0.0: Ligar (NA, botão de pulso)

I 0.1: Desligar (NF, botão de pulso)

Q 8.1: Planta ligada

Modo Manual / Automático

I 0.4: Automático / Manual

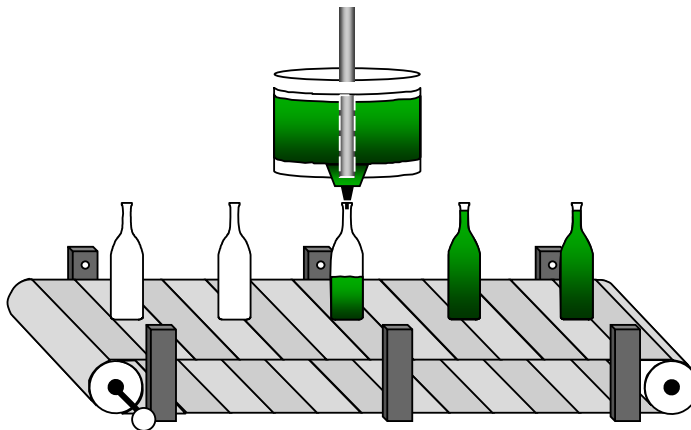
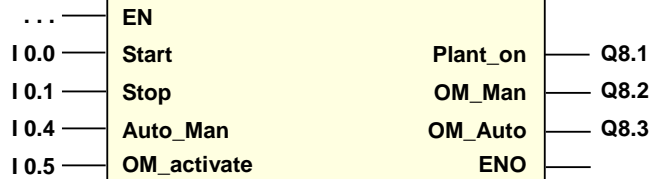
I 0.5: Confirma o modo

Q 8.2: Modo Manual selecionado

Q 8.3: Modo Automático selecionado

DB15

FB15: "Modo de seleção"



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.14



Conhecimento em Automação  
Training Center

### Nota

O controle de equipamentos de automação (acionamentos, transportador de correias, etc.) tomam lugar usando blocos de funções.

Para não infringir os princípios da programação estruturada, você não deve acessar diretamente endereços globais como entradas, saídas, memórias bit etc. dentro de um bloco de funções. Cada informação transferida com sinais do processo ou de outros blocos do programa do usuário deve tomar lugar usando os parâmetros do bloco.

Somente na chamada de um bloco no nível mais alto, isto é, no OB associado, você pode atribuir sinais de entrada do processo ou sinais de saída diretamente aos parâmetros do bloco.

### Controlando o modo: FB15

Você irá encontrar uma tabela de símbolos na pasta de programa

Chap\_06\_1 do projeto "Pro2\_ex51" para o exercício 6.1.

Antes de mais nada, criar um FB15 "Modo de Seleção", no qual a lógica completa de Liga/Desliga e a seleção do modo da planta é implementado.

FB15 "Modo de Seleção" tem os seguintes parâmetros de entradas e saídas:

#Start (in, BOOL): A planta é ligada usando #Start (1-ativo).

#Stop (in, BOOL): A planta é desligada usando #Stop (0-ativo). O parâmetro de entrada #Stop tem prioridade sobre #Start.

#Plant\_on (out, BOOL): Quando a planta é ligada #Plant\_on é ajustada para "1".

Você pode selecionar o modo quando a planta é ligada.

#Auto\_Man (in, BOOL): #Auto\_Man=0 seleciona modo manual  
#Auto\_Man=1 seleciona modo automático

#OM\_activate (in, BOOL): O modo selecionado é confirmado quando existe um pulso em #OM\_activate.

O modo ativo é indicado pelo FB15 nos seguintes parâmetros de saída :

#OM\_Man (out, BOOL): modo manual está ativo

#OM\_Auto (out, BOOL): modo automático está ativo



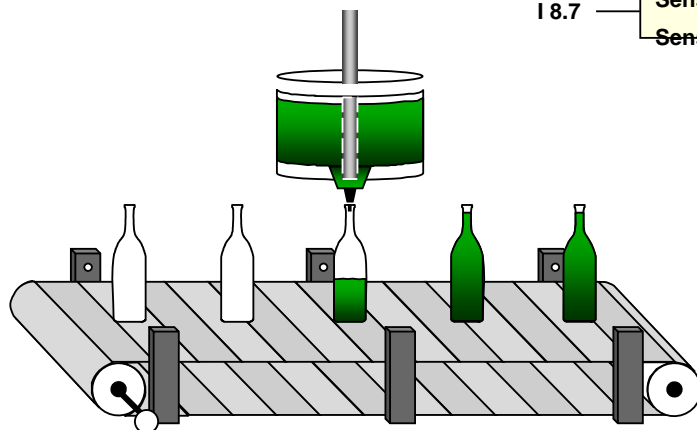
## Exercício 6.1b: Planta de Engarrafamento - Transportador

### Modo Manual

I 0.5: Jog p/frente  
 I 0.6: Jog p/trás  
 Q 8.5: Transportador p/frente  
 Q 8.6: Transportador p/trás

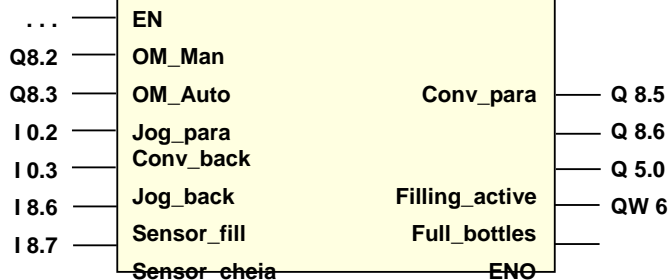
### Modo Automático

I 8.6: Sensor: local enchimento  
 I 8.7: Sensor: contagem garrafas  
 Q 5.0: Enchimento ativo  
 QW 6:: Mostra garrafas cheias



DB16

### FB16: "Controle Transportador"



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
 File: PRO2\_06P.15



Conhecimento em Automação  
 Training Center

### Controle transportador: FB16

Criar um FB16 "Controle Transportador", contendo a lógica completa para o controle do transportador em modo manual e automático.

FB16 tem os seguintes parâmetros de entrada e saída :

#OM\_Man (in, BOOL): Transportador é operado no modo manual.

#OM\_Auto (in, BOOL): Transportador é operado no modo automático.

#Jog\_para (in, BOOL): O transportador pode ser movimentado para frente no modo manual usando esta entrada. Esta entrada é sem importância no modo modo automático.

#Jog\_back (in, BOOL): O transportador pode ser movimentado para trás no modo manual usando esta entrada. Esta entrada é sem importância no modo modo automático.

#Sensor\_fill (in, BOOL): Indica que uma garrafa vazia se encontra posição enchimento.

#Sensor\_cheia (in, BOOL): Indica que outra garrafa cheia passou pela barreira de contagem garrafas cheias.

#Conv\_para (out, BOOL): Entrega o sinal de controle para operação do transportador p/frente.

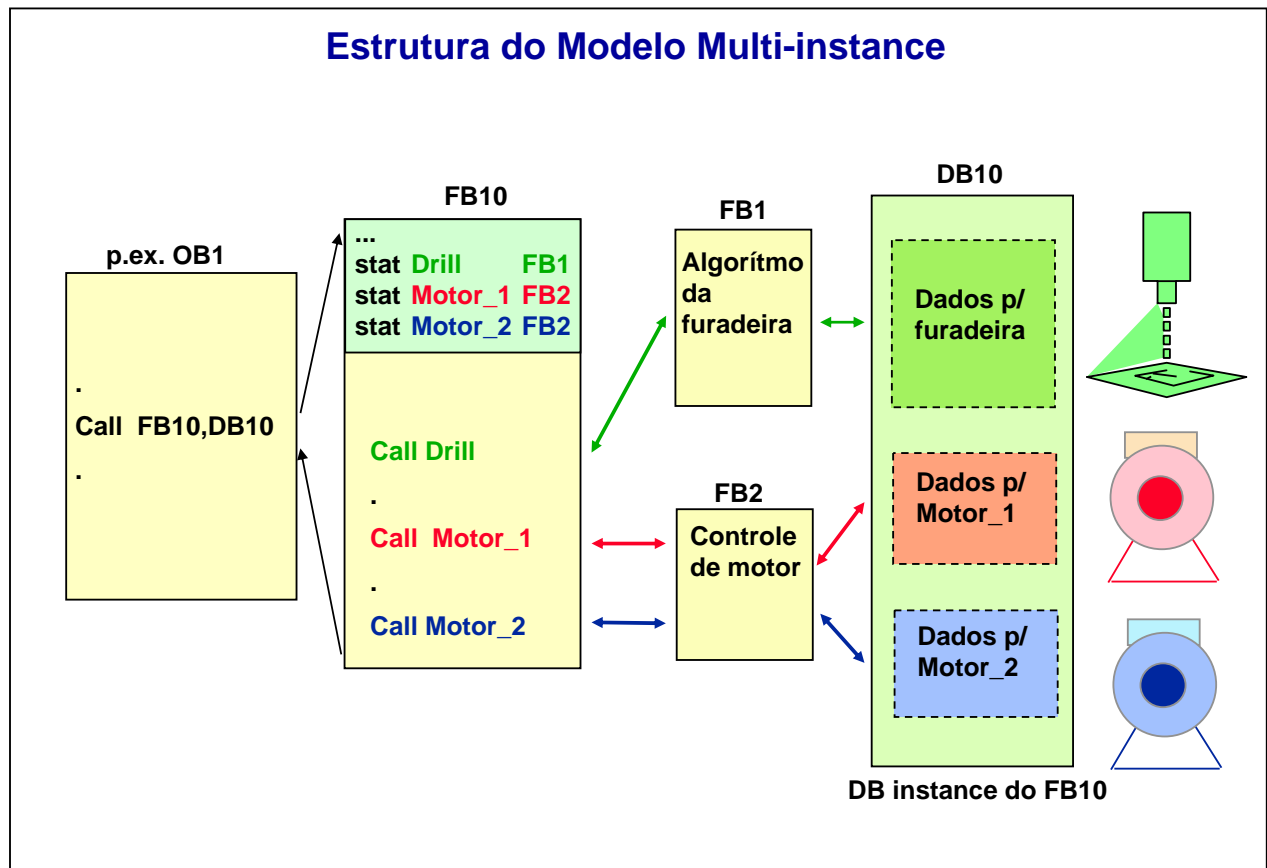
#Conv\_back (out, BOOL): Entrega o sinal de controle para operação do transportador p/trás.

#Filling\_active (out, BOOL): Indica que o enchimento está ativo.

#Full\_bottles (out, WORD): Dá o número de garrafas cheias em formato BCD.

Chamada de ambos os blocos com o DB instance associado DB15 e DB16 no OB1 e atribuir os parâmetros dos FBs com os sinais do painel de operação (simulador) ou com os sinais do processo (transportador) conforme as figuras acima.

## Estrutura do Modelo Multi-instance



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.16Conhecimento em Automação  
Training Center

### Modelo Multi-instance

Adicionalmente a atribuição de valores de blocos de funções pela especificação de um DB instance na chamada de um FB, o STEP 7 também suporta a declaração explícita de FBs instances dentro de um bloco de funções de alto nível.

Por isto, instances de chamadas de blocos de funções são declaradas com tipos de dados FB1 ou FB2 usando identificadores simbólicos (Drill, Motor\_1 e Motor\_2), na seção de declaração da chamada do bloco de funções FB 10 na seção "variáveis estáticas". Dentro de bloco de funções de alto nível, os instances individuais são então chamados usando seu identificador simbólico. O bloco de funções de alto nível FB10 deve portanto ser chamado com com seu próprio DB instance (DB10).

O STEP 7 assegura, na criação de DB instance de alto nível, que as áreas de dados necessárias para os instances individuais sejam atribuídas na área de dados do FB10 de alto nível.

Na chamada dos instances individuais usando os nomes simbólicos, a chamada macro assegura que o registrador AR2 seja ajustado no início da área de dados atribuídos ao instance de forma que os parâmetros e variáveis locais do instance sejam também acessados durante o processamento do chamado bloco de funções.

### Vantagens

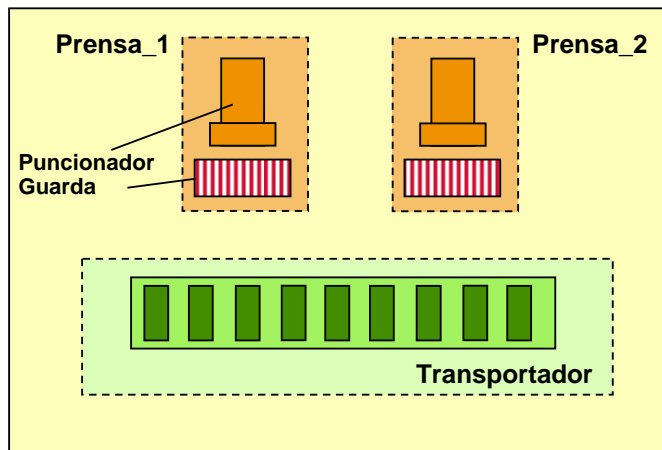
The utilização do modelo multi-instance tem as seguintes vantagens:

- Os instances individuais não requerem seu próprio bloco de dados a cada vez. Dentro de uma chamada hierárquica dos blocos de funções, somente um DB instance é "vestido" na chamada do bloco de funções "mais externo".
- O modelo multi-instance "solda" um bloco de funções e uma área de dados instance dentro de um mesmo objeto (FB instance), que pode também ser manipulado como uma unidade. O programador não precisa se preocupar com o gerenciamento (criação, endereçamento) das áreas de dados instance individuais. Ele deve simplesmente fornecer um DB instance para o FB "mais externo".
- O modelo multi-instance suporta o estilo de programação orientada a objeto.

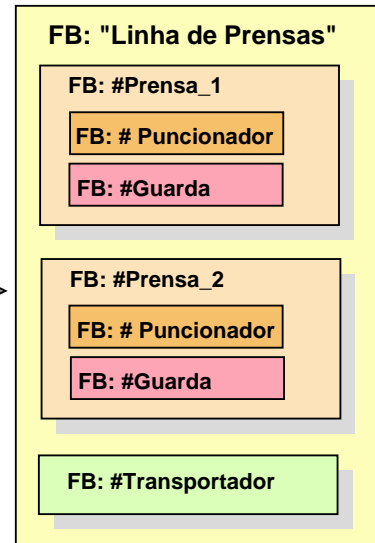


## Programação Orientada a Objeto usando Multi-instances

### Exemplo: Linha de Prensas



Divisão Tecnológica



Divisão Tecnológica do programa com a ajuda de FB instance

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.17



Conhecimento em Automação  
Training Center

### Unidades do processo

Unidades do processo são objetos físicos do processo, como as unidades de uma linha de montagem (correia transportadora, estações de processamento) ou uma máquina completa ou partes de uma máquina (p.ex. uma prensa completa ou o punçador ou a guarda de uma prensa).

Unidades de processo são usadas para critério de identificação lógica. Eles são, como regra, de um projeto hierárquico. Deste modo, unidades de processo podem, desta forma, conter sub-unidades (e.g. a unidade "prensa" contem as unidades "punçador" e "guarda"). Unidades de processo podem in deste modo serem projetadas para mais detalhadas sub-unidades. (Agregação).

### Estilo de programação orientada a objeto

Você pode implementar um estilo de programação orientada a objeto com a ajuda de blocos de funções.

A descrição técnica de uma unidade de processo ou programa de processo de sub-unidade é feito com um FB instance. A divisão do programa do usuário em unidades é arquivada pela declaração FB instances de baixo nível dentro de um FB de alto nível.

Desta forma, a mesma divisão dentro de unidades de processo é arquivada no programa do usuário como no sistema existente ou máquina. (Conceito de programação orientada a objeto agregação).

### Reutilização de Software

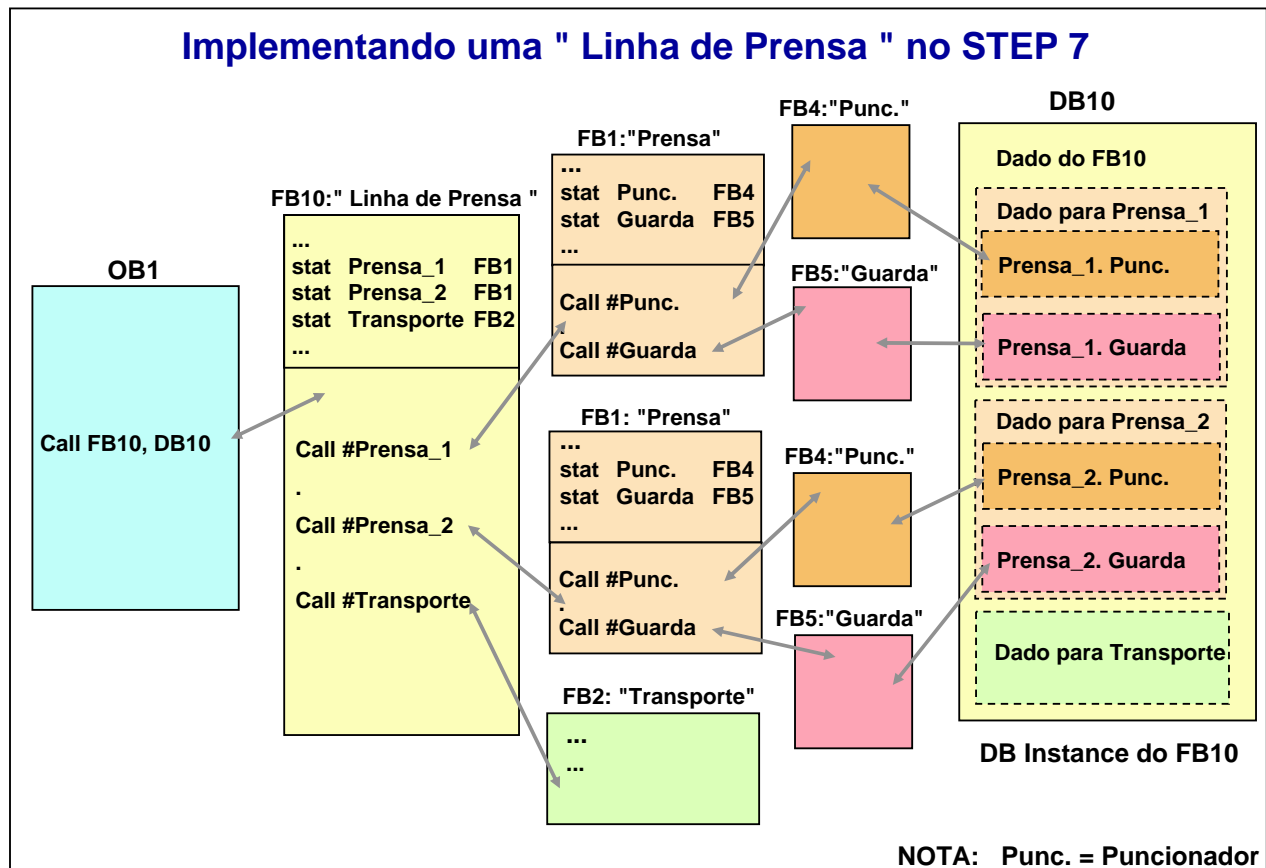
Este conceito hierárquico oferece uma alta disponibilidade para reutilização

de software gerado uma única vez e portanto oferece um grande potencial econômico na criação, modificação e manutenção de programas do usuário:

- Toda vez que um fabricante cria uma sub-unidade de processo (válvula, motor, etc.), ele também entrega um FB para controle desta sub-unidade de processo.
- Toda vez que uma unidade de processo físico é construída dentro de uma grande unidade, um FB instance da unidade também é declarada no FB da unidade de alto nível.

FBs são os componentes básicos de controle de programa. Em estágios de planejamento de um programa, os FBs tem a mesma tarefa que os circuitos integrados (CIs) em placas de circuito impresso de fabricantes. A estrutura do programa do usuário consiste de FBs pré-fabricados, que devem simplesmente ser interconectados.

## Implementando uma " Linha de Prensa " no STEP 7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.18



Conhecimento em Automação  
Training Center

### Modelo

#### Multi-instance

Quando um modelo multi-instance é usado, o DB instance contém os dados para diversos blocos de funções em uma chamada hierárquica. Para isto, os instances dos FBs chamados são declarados usando identificadores simbólicos como variável *stat* na seção de declarações do FB chamado.

O FB instance "mais externo" para a unidade de processo (neste caso: FB10 "Linha de Prensa") deve ser chamado absolutamente ou simbolicamente acompanhado pela especificação de seu próprio DB instance (neste caso: DB10).

### Declarações

Como variáveis *stat* da seção de declarações do FB10 ("Linha de Prensa"), duas variáveis instances do tipo de dado FB1 ("Prensa") com os nomes *#Prensa\_1* e *#Prensa\_2* foram declaradas, bem como também um instance do tipo de dado FB2 ("Transporte") com o nome *#Transporte*.

Na seção de declarações do FB1 ambos, um instance do FB4 ("Puncionador") com the nome *#Puncionador* e um instance do FB5 ("Guarda") com the nome *#Guarda*, foram declaradas.

Na seção de instrução do FB1 ("Prensa"), os respectivos FBs instances são então chamados usando os nomes simbólicos *#Puncionador* e *#Guarda* que estavam declarados na seção de declaração.

### Nota

A declaração de um instance na seção de declaração de um bloco de funções trabalha somente se o FB, do qual um instance está sendo declarado, já existe.

Quando projetado como uma chamada hierárquica, estes FBs que estão sendo chamados por último devem ser criados primeiro.

### Multi-instance DB

The multi-instance DB tem the same structure as the declaração parts of the funções blocos concerned. se an instance é chamado no instrução seção, então it automatically acessado the dado no corresponding seção of the instance DB (DB10).

## Propriedades do Modelo Multi-instance

### Vantagens do modelo Multi-instance :

- Somente um DB é necessário para diversos instances.
- Nenhum gerenciamento adicional é necessário na montagem das áreas de dados "privados" para os respectivos instances.
- O modelo multi-instance opera como "estilo de programação orientada a objeto" (reutilização por meio de "Agregação").
- Maximum nesting depth of 8

### Pré-requisitos para os FBs:

- Acesso direto aos sinais do processo (I, Q) dentro do FB é não possível.
- Acesso aos sinais do processo ou comunicação com outras unidades de processo podem somente tomar lugar usando parâmetros do FB.
- O FB pode somente "lembrar" estados do processo in suas variáveis estáticas, não em DBs globais ou memórias bit (M).

### Nota:

- Dado instance também pode ser acessado do „lado de fora" p.ex. no OB1: L "Linha de Prensa".Prensa\_2.Puncionador.<VarName>

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.19Conhecimento em Automação  
Training Center

### Benefícios do modelo Multi-instance

Com o modelo multi-instance você pode armazenar as respectivas seções de dados de diversos instances de um e a mesma chamada hierárquica em um simples DB.

Desta forma somente um DB é necessário para diversos instances.

Com o modelo multi-instance nenhuma medida para a administração de dados locais no FB são necessárias, exceto para a atribuição de um DB instance comum.

O modelo multi-instance suporta o conceito de programação orientada a objeto. Códigos e dados, que são necessários para o controle de unidades de processo são agregados em FBs.

Se uma unidade de processo consiste de sub unidades hierárquicas então exatamente esta estrutura pode ser refletida no programa do usuário através do modelo multi-instance.

O programa de controle pode ser projetado com FBs instances do mesmo modo que as máquinas são constituídas de componentes.

O STEP7 suporta uma dimensão de aninhamento de 8 com o modelo multi-instance.

### Pré-requisitos para Multi-instances

De forma a utilizar um FB como multi-instance sem problemas, os seguintes pontos devem ser atendidos:

- Para o propósito de controle de processo, nenhum acesso direto a endereços globais da CPU (como as entradas e saídas) são permitidos. O acesso a entradas e saídas globais impossibilitam a reutilização.
  - Comunicação com o processo ou com outras seções de programa (FBs) deve somente ser feito usando FB parametrizáveis.
- Somente após a integração do FB dentro de uma unidade de alto nível, é a "atribuição" do FB através da lista de parâmetros executa com a chamada do FB.
- Estados ou outras informações sobre a unidade a ser controlada devem ser "lembradas" pelo FB em suas próprias variáveis estáticas.

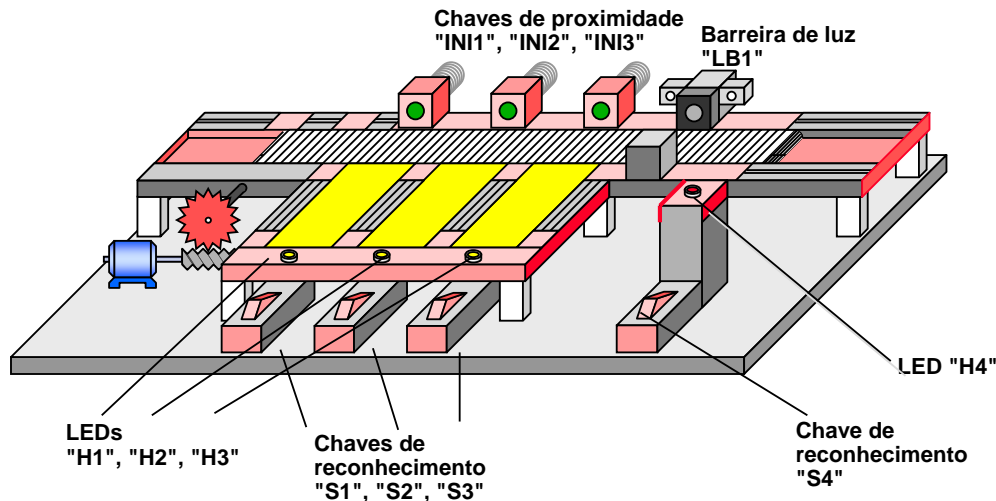
## Exercício 6.2: O Modelo Transportador como Linha de Montagem

### Seqüência do processo para a estação de trabalho

- Processamento da peça
- Processamento terminado
- Lugar da peça na correia
- Espera pela peça bruta
- Pega peça bruta da correia

### Seqüência de processo para a correia transportadora

- Espera para término da peça
- Transporte para montagem final
- Montagem final, insere peça bruta
- Transporte para estação



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.20



Conhecimento em Automação  
Training Center

### Objetivo

Por meio de uma linha de montagem, o princípio de solução da tarefa está sendo praticado usando programação de FBs. Um FB separado é usado em cada caso para o controle da estação de trabalho 1 e a correia transportadora. O FB para a estação de trabalho estará funcionando como multi-instance.

No próximo exercício, a funcionalidade da linha de montagem deverá ser expandida para as estações de trabalho 2 e 3 por meio do modelo multi-instance.

### Princípio das funções do modelo transportador

Para os exercícios de programação de FBs, o modelo transportador é para operar como em uma linha de montagem com as seguintes funcionalidades (por enquanto, somente uma estação de trabalho):

1. O sistema está no estado inicial, isto é, a estação de trabalho 1 tem uma peça que está sendo processada no momento. Isto é indicado com um LED "H1" pisca lento na localização 1.

A correia transportadora não está ocupada, ou seja, não há peça em "INI1" nem na montagem final "LB1". O motor da correia está desligado.

2. Após a peça ter sido terminada, o operador dá o reconhecimento disto com a chave de reconhecimento "S1". O LED "H1" pisca rápido.

3. O operador coloca a peça terminada na correia "vazia" em frente a chave de proximidade "INI1". O LED "H1" desliga.

4. A correia então transporta a peça terminada para a montagem final.

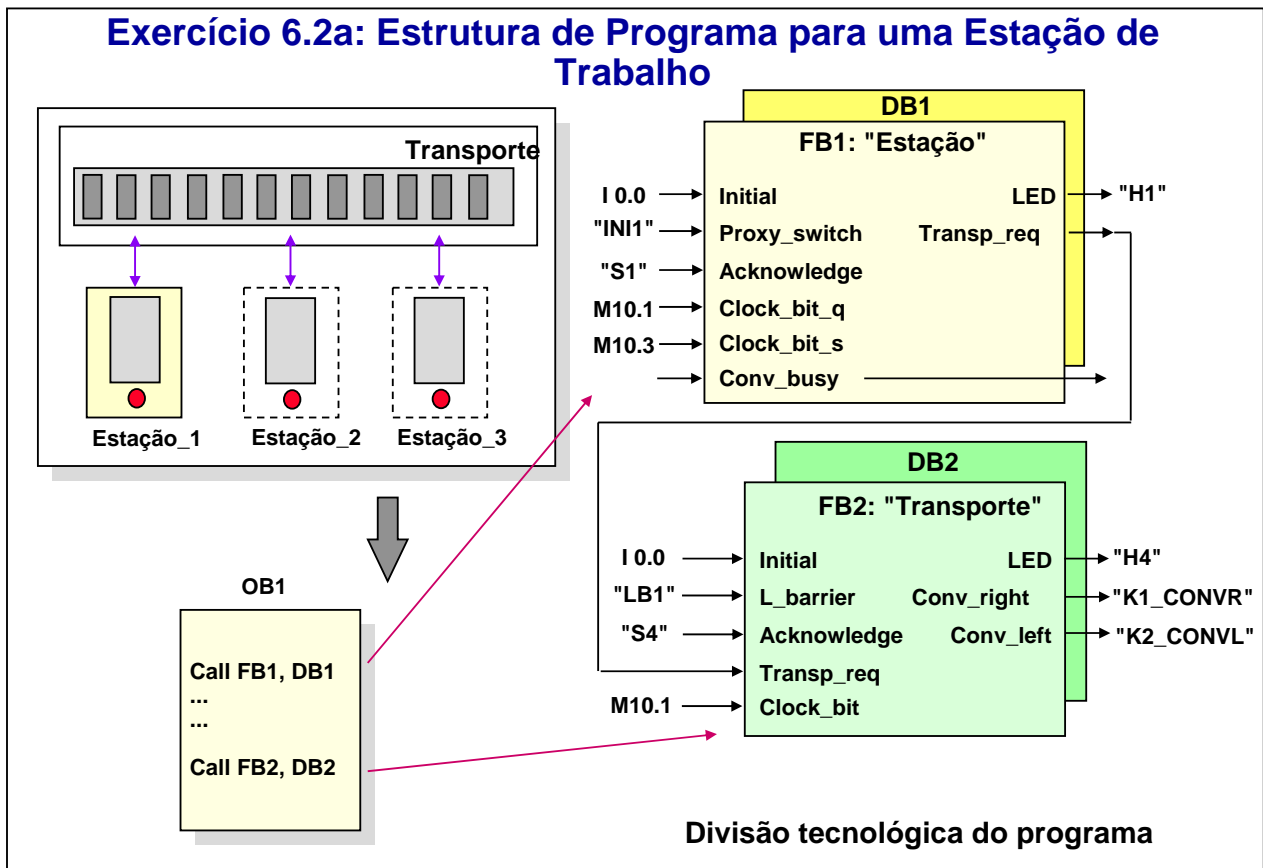
O LED "H4" pisca rápido durante o transporte. Quando a posição final de montagem é alcançada, o LED "H4" fica aceso.

5. O operador na montagem final pega a peça terminada da correia e coloca a nova peça bruta na correia. Ele então dá o reconhecimento disto com a chave "S4".

6. A correia transporta a nova peça bruta de volta para a estação de trabalho 1. O LED "H4" pisca rápido durante o transporte. Quando a chave de proximidade "INI1" é alcançada, o LED "H1" da estação de trabalho começa a piscar rápido.

7. O operador pode pegar a peça bruta da correia e colocá-la na estação de trabalho 1 e recomençar o processo novamente. O LED "H4" pisca lento novamente. O processo de trabalho recomeça com Step 1.

## Exercício 6.2a: Estrutura de Programa para uma Estação de Trabalho



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.21Conhecimento em Automação  
Training Center

### Tarefa

Na primeira etapa, somente uma estação de trabalho da linha de montagem está sendo implementada. O controle do sistema em questão é desta forma dividido em duas unidades de processo :

- Estação: controle da primeira estação de trabalho, implementado pelo FB1 com o nome simbólico global "estação" (tabela de símbolos globais).
- Transporte: controle da correia transportadora, implementado pelo FB2 com o nome simbólico global "transporte"

Para o controle total da linha de montagem, ambos FBs são então chamados no OB1 cada um com seu próprio DB instance.

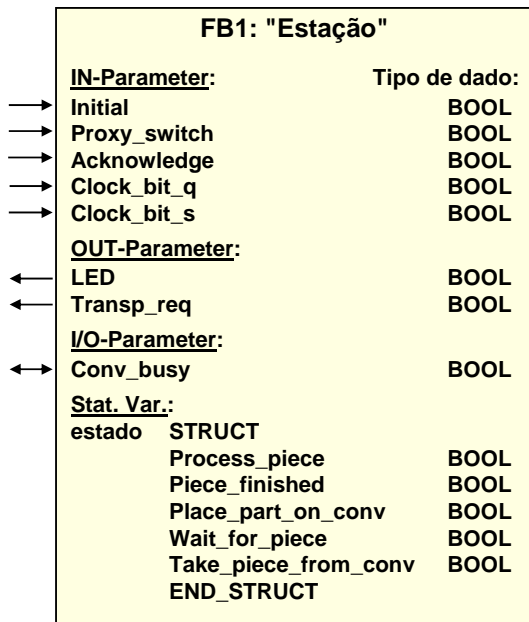
### O que fazer

Você irá encontrar os correspondentes blocos de funções FB1 e FB2 bem como a respectiva tabela de símbolos na pasta de programa Chap\_06\_2 do projeto "Pro2\_e\_x51" para o exercício 6.2.

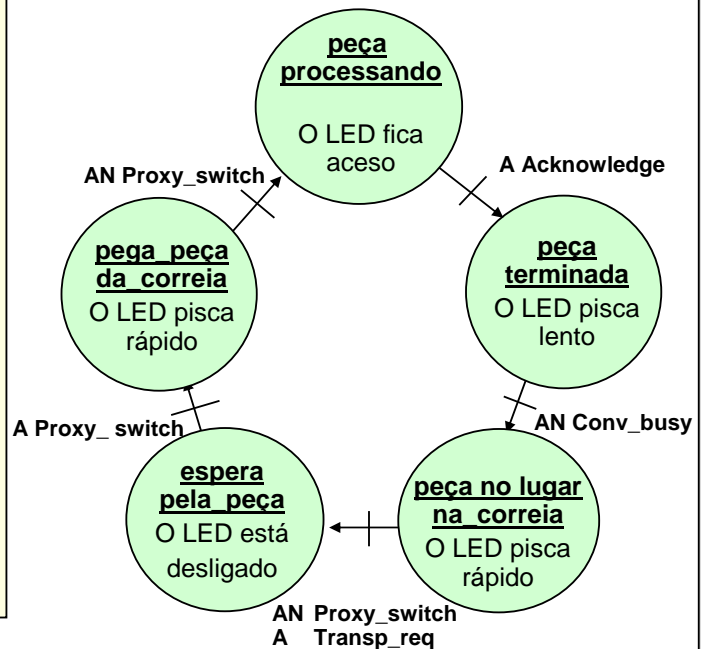
1. Abra o projeto "Pro2\_e\_x51" e insira a configuração de hardware da sua estação de treinamento.
2. Para a CPU, atribuir parâmetros para um bit de clock no MB10 e então transfira a configuração de hardware para dentro da CPU da estação de treinamento.
3. Chamar o FB1 com o DB instance DB1 no OB1 e atribuir os sinais de processo para a interface de parâmetros. Atribuir os parâmetros de entrada #Clock\_bit\_q com M10.1 e #Clock\_bit\_s com M10.3 .
4. Programar a chamada do FB2 (DB instance DB2) após a chamada do FB1 no OB1. Atribuir os parâmetros de entrada e saída de acordo para a descrição do FB1 e FB2.
5. Transferir os blocos para a CPU e testar a funcionalidade do seu programa.

## FB1 "Estação" – Método de Funcionamento

### ● Declarações no FB1:



### ● Modelo de estado:



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.22Conhecimento em Automação  
Training Center

#### Inicialização

Com um impulso no parâmetro de entrada *#Initial*, FB1 pode ser inicializados com o estado *#Process\_piece*. Atribuir *#Initial* com I 0.0.

#### *#Process\_piece*

A peça é processada neste estado. O LED "H1" está continuamente aceso para indicar o processamento.

Uma transição para o estado *#Piece\_finished* ocorre quando o operador reconhece o término da peça com a chave "S1".

#### *#Piece\_finished*

Neste estado o LED pisca com a frequência do *#Clock\_bit\_s* (frequência pisca lento). O operador aguarda pela correia transportadora para ser habilitada (Importante para a expansão para 3 estações!).

Se o transportador está vazio (*#Conv\_busy*=0), ele é imediatamente definido como ocupado (*Conv\_busy*=1) e vai para o estado *#Place\_piece\_on\_Conv*.

#### *#Place\_piece\_on\_conv*

Neste estado - o LED pisca com a frequência do *#Clock\_bit\_q* (frequência pisca rápido) - o operador pode colocar a peça no transportador. Com o sinal *#Proxy\_switch*=1, o sinal *#Transp\_req*=1 também é setado, então inicia o movimento do transportador para a posição final de montagem.

Uma transição para o estado *#Wait\_for\_piece* ocorre quando a peça deixa a chave de proximidade (*#Proxy\_switch*=0).

#### *#Wait\_for\_piece*

O operador espera para a chegada de uma nova peça bruta neste estado; o LED em frente da estação está desligado.

Com the chegada de uma nova peça bruta (*Proxy\_switch* =1), a transição para o estado *#Take\_part\_from\_conv* toma lugar.

#### *#Take\_piece\_from\_conv*

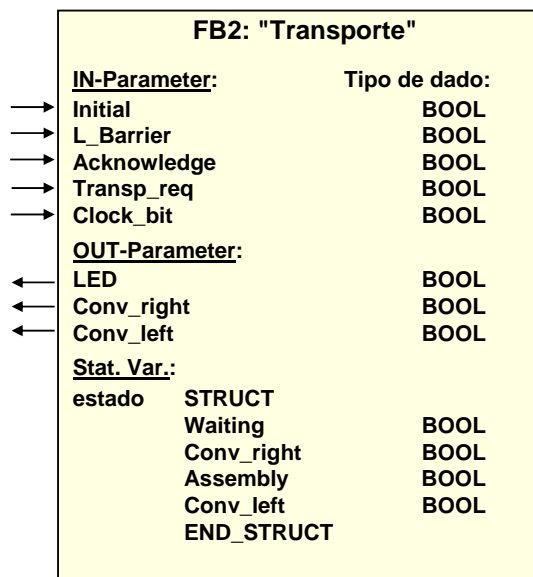
Neste estado - o LED pisca com *#Clock\_bit\_q* (frequência pisca rápido) - a peça pode ser pega do transportador.

Uma transição para o estado *#Process\_piece* toma lugar quando a peça está acessível (*#Proxy\_switch* =0).

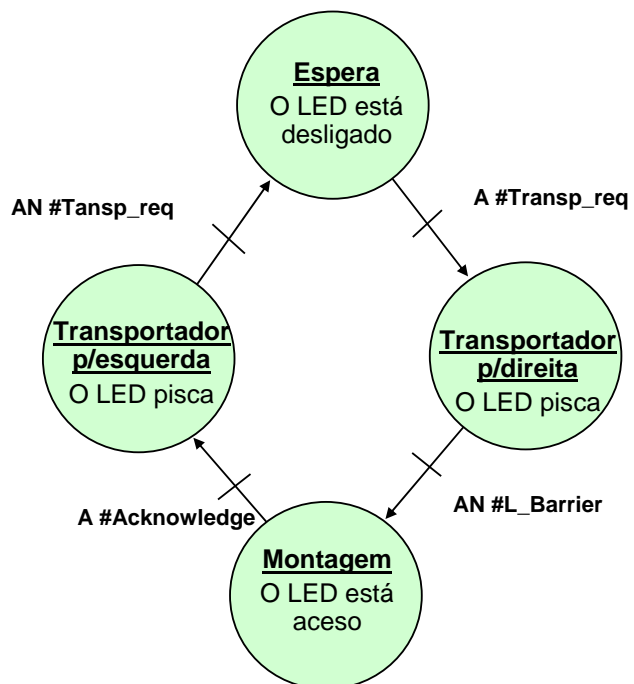


## FB2 "Transporte" – Método de Funcionamento

### Interface do FB2:



### Modelo de estado:



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.23Conhecimento em Automação  
Training Center

### Inicialização

O FB2 pode ser inicializado com o estado *#Waiting* através do sinal de entrada *#Initial*. Atribui o parâmetro de entrada *#Initial* com I 0.0.

### #Waiting

Neste estado, a correia transportadora espera pelo término da peça que é colocada no transportador por uma das estações. Durante o tempo que a correia transportadora está no estado *#Waiting*, ela fica parada e o LED "H4" está desligado.

Com o estado 1 o sinal *#Transport\_req*, uma transição para o estado *#Conv\_right* toma lugar.

### #Conv\_right

Neste estado, a peça é transportada na direção da montagem final. Durante o tempo que o transportador é movimentado, o LED "H4" pisca com a frequência (M10.1) dado pelo parâmetro de entrada.

A montagem final é alcançada, isto é, a chave para o estado *#Assembly* toma lugar, quando a peça terminada passar pela barreira de luz "LB1".

### #Assembly

Neste estado, o operador troca a peça terminada com uma nova peça bruta. O LED "H4" fica aceso neste estado. O operador sinaliza o término desta tarefa com a chave "S4".

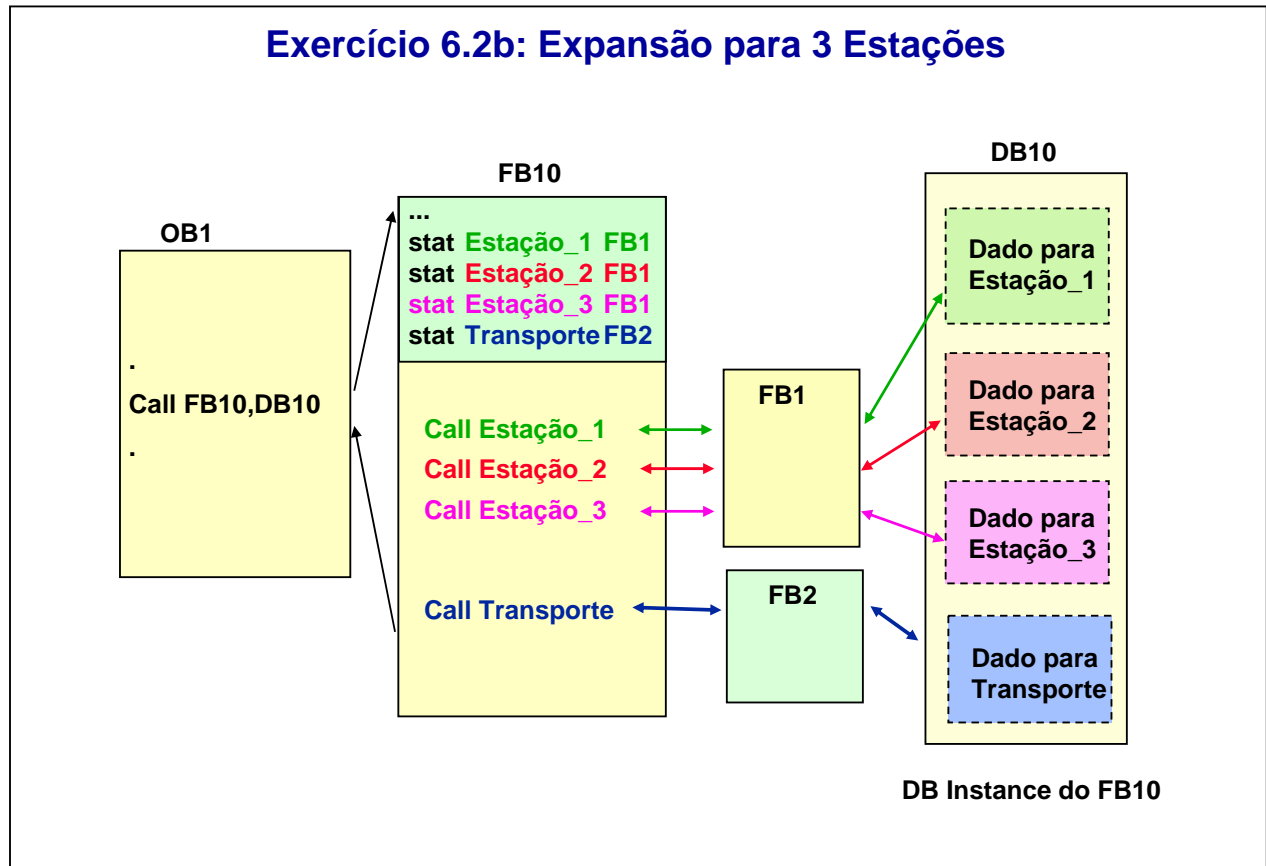
Este sinal também comanda para a transição para o estado *#Conv\_left*.

### #Conv\_left

Neste estado, a peça bruta é transportada na direção da estação de trabalho. Durante o tempo que o transportador é movimentado, o LED "H4" pisca com a frequência dada pelo parâmetro de entrada *#Clock\_bit*.

O transportador é parado quando o sinal de entrada *#Transp\_req* é resetado. Uma transição para o estado *#Waiting* também toma lugar.

## Exercício 6.2b: Expansão para 3 Estações



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.24



Conhecimento em Automação  
Training Center

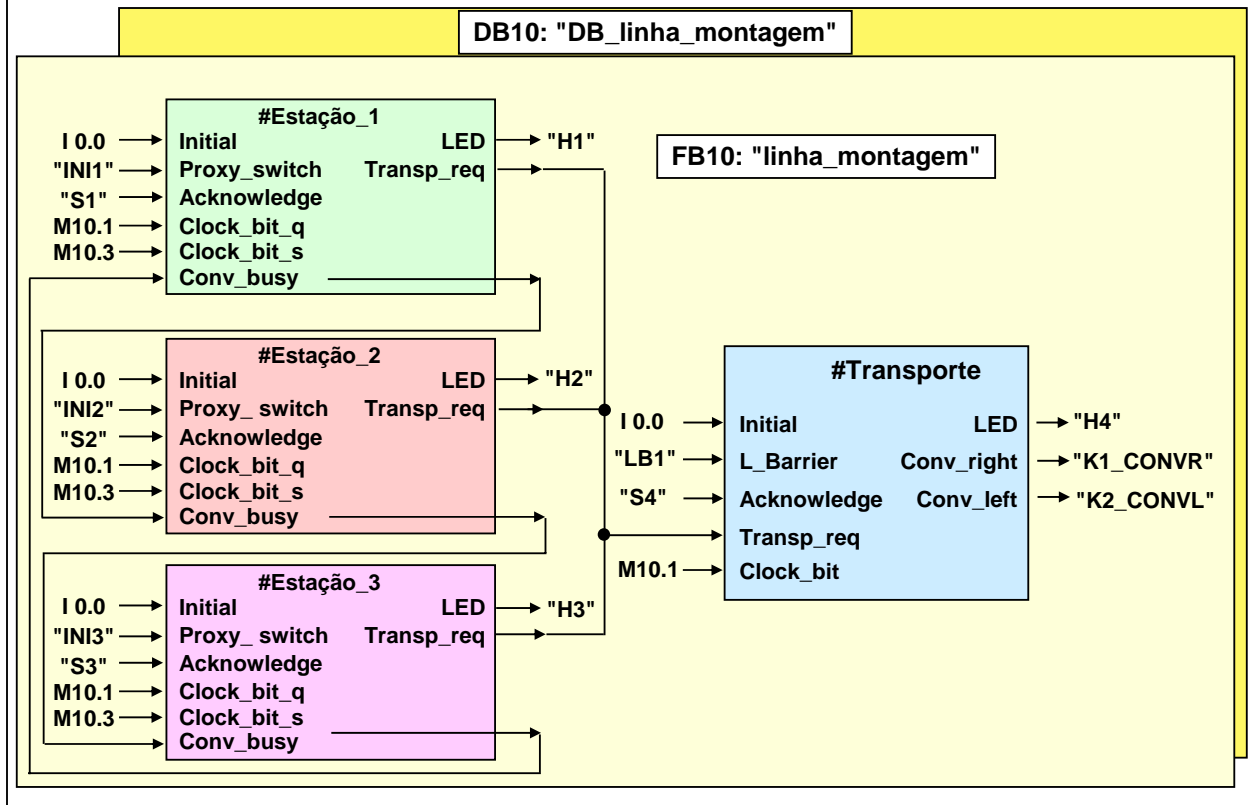
### Estrutura do programa

Na segunda parte deste exercício, a funcionalidade total do modelo transportador para todas as três estações de trabalho está sendo finalizada. Para isto, o controle total do modelo transportador (3 estações e uma correia transportadora) está sendo movida para dentro de um único FB (FB10).

Dentro do FB10, o controle das três estações de trabalho é implementado como instances separados do FB1 e o controle da correia transportadora estão implementados como instance do FB2.



## Interconexão de Parâmetros de Blocos



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_06P.25



Conhecimento em Automação  
Training Center

### O que fazer

1. Primeiro de tudo criar um FB10. Na seção *stat. Var.*, declare os três instances of FB1 com os nomes: **#Estação\_1**, **#Estação\_2** e **#Estação\_3** e um instance do FB2 com o nome: **#Transporte**.
2. No FB10, primeiro de tudo, chamar na sequência **#Estação\_1**, **#Estação\_2**, **#Estação\_3** e **#Transporte** e interconectar os parâmetros dos blocos correspondentemente para o esquema acima.

Note a interconexão do parâmetro in/out **#Conv\_busy**. Como este pode ser implementado? Podem ser usadas aqui as variáveis auxiliares temporárias ou estáticas?

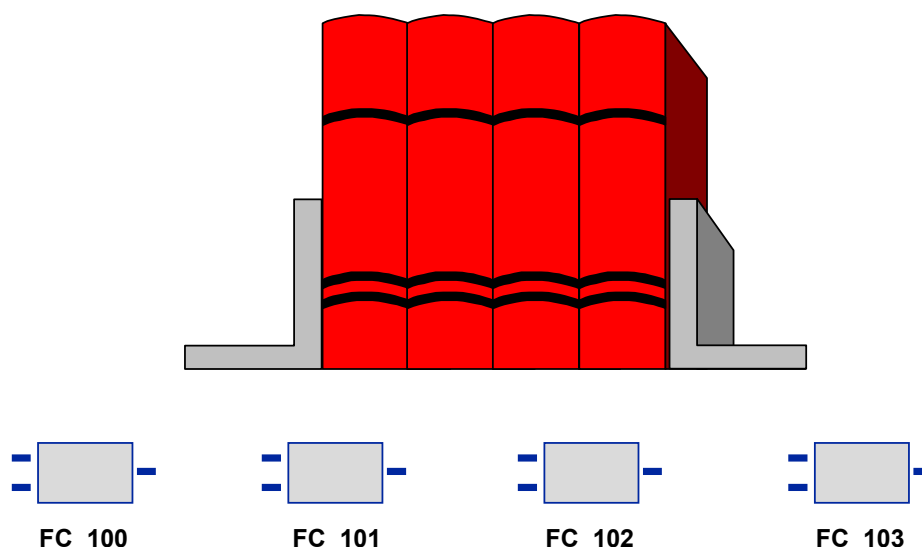
Também preste atenção para a interconexão do parâmetro de saída **#Transp\_req** (lógica OU) para o parâmetro de entrada **#Transp\_req** do controle da correia. Como pode ser implementada como uma interconexão?

3. Criar um DB10 claramente vinculado com o FB10. Edite o DB10 e verifique sua estrutura na declaração e "data view" do Editor de DB.
4. Chamar o FB10 com o DB instance DB10 no OB1.
5. Transfira os blocos participantes para a CPU e teste o resultado.

### Perguntas

- Quais são as vantagens e desvantagens deste tipo de solução?
- Como poderá o controle ser expandido, de forma que uma linha de montagem "vazia" pode também ser "enchida" ou uma linha "cheia" também pode ser "esvaziada".

## Utilizando Bibliotecas



SIMATIC S7

Siemens AG 1999. All rights reserved.

data: 04.10.2007  
File: PRO2\_07P.1



Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Fatos interessantes sobre Bibliotecas .....	2
Configuração e Conteúdo da Biblioteca Padrão .....	3
Fatos interessantes sobre Funções do Sistema (SFC) .....	4
Vista geral das Funções do Sistema (SFC) (Parte 1) .....	5
Vista geral das Funções do Sistema (SFC) (Parte 2) .....	6
Vista geral das Funções do Sistema (SFC) (Parte 3) .....	7
Vista geral das Funções do Sistema (SFC) (Parte 4) .....	8
Vista geral das Funções do Sistema (SFC) (Parte 5) .....	9
Chamada das Funções do Sistema (SFC) e Blocos de Funções do Sistema (SFB) .....	10
Avaliação de uma Mensagem de Erro .....	11
Exercício 7.1: Geração de um DB com um atributo "UNLINKED" .....	12
Exercício 7.2: Testando um Bloco de Dados (SFC 24: somente para S7-400) .....	13
Exercício 7.3: Geração de um DB (SFC 22) .....	14
Exercício 7.4: Copiando um DB da Memória de Carga para a Memória de Trabalho (SFC 20) .....	15
Exercício adicional 7.5: Inicializando um DB (SFC 21) .....	16
Exercício adicional 7.6: Escrevendo uma mensagem no Buffer de Diagnóstico (SFC 52) .....	17
Exercício adicional 7.7: Bloco Contador com função "Debouncing de Contato" .....	18
A Biblioteca: Conversão de Blocos S5-S7 .....	19
A Biblioteca: Conversão de Blocos TI-S7 (Parte 1) .....	20
A Biblioteca: Conversão de Blocos TI-S7 (Parte 2) .....	21
A Biblioteca: Blocos de Comunicação .....	22
A Biblioteca: Blocos de Controle PID .....	23

## Fatos interessantes sobre Bibliotecas

### Propósito:

- Arquivamento de componentes de programa reutilizáveis
- Transferência direta para a CPU e teste não é possível

### Configuração da Biblioteca :

- A biblioteca pode conter diversas pastas de programa
- A biblioteca não pode conter qualquer "Hardware"
- Cada pasta de programa contem:
  - As pastas "Blocks", "fonte Files", "Symbols"
  - A pasta "Charts" (somente para a opção de software: S7-CFC)

### Uso das Bibliotecas:

- Com o SIMATIC Manager:
  - Bibliotecas podem ser nomeadas (mas não com os mesmos nomes dos Projetos)
  - Blocos podem ser copiados entre bibliotecas e projetos
  - Bibliotecas podem ser arquivadas

## SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.2



Conhecimento em Automação  
Training Center

### Vista Geral

Bibliotecas são usadas para guardar componentes de programa reutilizáveis para o SIMATIC S7/M7. Os componentes de programa podem ser copiados dos projetos existentes para dentro de uma biblioteca ou eles podem ser gerados diretamente na biblioteca independentemente dos projetos.

A mesma funcionalidade dos projetos é disponível para a geração de programas S7 em uma biblioteca com a exceção dos testes.

### Configuração

Exatamente como os projetos, as bibliotecas são configuradas de uma maneira hierárquica:

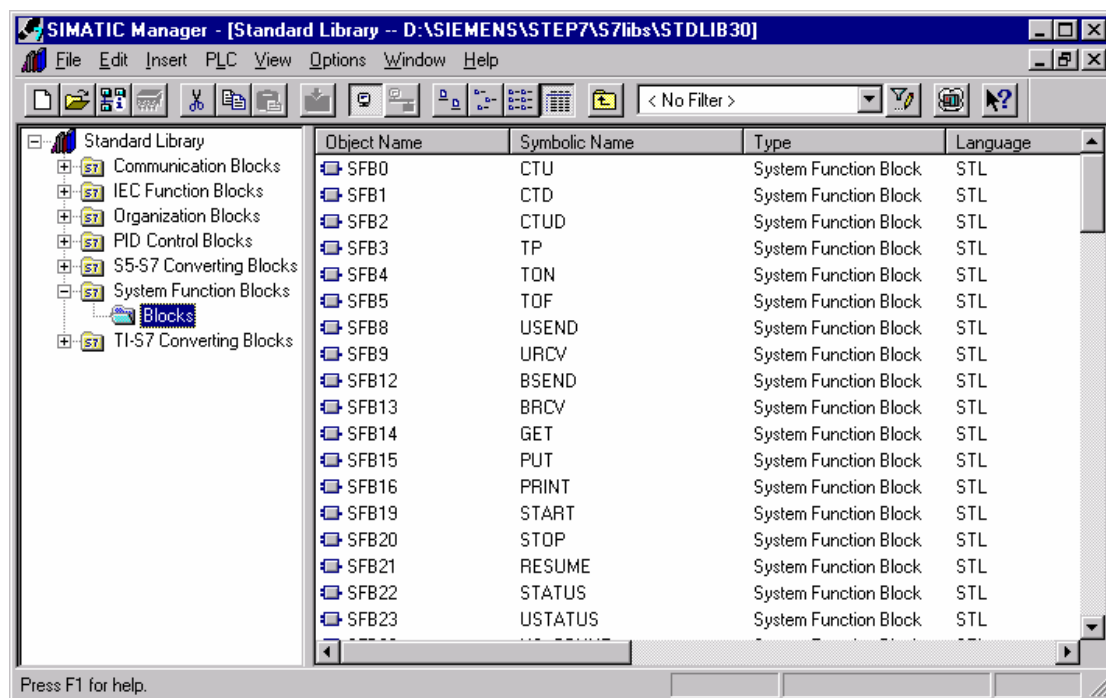
- Bibliotecas podem conter programas S7.
- Um programa S7 pode conter exatamente uma pasta *Blocks*, uma pasta *fonte Files*, uma pasta *Charts* bem como também um objeto *Symbols* (tabela de símbolos).
- A pasta *Blocks* contem os blocos, que podem ser carregados para dentro da CPU S7. A tabela de variáveis (VATs) e o tipo de dado definido pelo usuário (UDTs) contidos nela não são carregáveis para dentro da CPU.
- A pasta *fonte Files* contem as fontes para os programas gerados nas diversas linguagens de programação.
- A pasta *Charts* contem os CFC-Charts (somente para a opção de software S7-CFC).

Quando você insere um novo programa S7, uma pasta *Blocks* e uma pasta *fonte Files* bem como também um objeto *Symbols* são automaticamente criados nela.

### Uso das Bibliotecas

Blocos que são usados repetidas vezes podem ser guardados em bibliotecas. De lá eles podem ser copiados para dentro do programa do usuário e serem chamados por outros blocos.

## Configuração e Conteúdo da Biblioteca Padrão



### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.3



Conhecimento em Automação  
Training Center

### Introdução

Duas bibliotecas padrão são automaticamente instaladas no disco rígido com a instalação do software STEP7:

- A biblioteca padrão **stdlibs**(V2) para Versão 2 e
- A biblioteca **standard** V3.x para Versão 3.

Destas bibliotecas você pode copiar os blocos desejados para o seu projeto.

### Abrindo uma Biblioteca

Para abrir uma biblioteca, utilize os seguintes comandos: *File -> Open* ou os ícones associados na barra de ferramentas.

Um diálogo subsequente é aberto no qual você pode selecionar o projeto desejado ou a biblioteca desejada.

### Biblioteca Padrão

A biblioteca padrão Standard Library V3.x contém os seguintes programas S7:

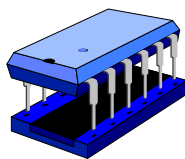
- comunicação Blocks: contêm as funções para conexão com I/O distribuído quando se utiliza uma CP Profibus S7-300.
- IEC Converting Blocks: contêm blocos para funções IEC p.ex. para manipulação de tipos de dados dado\_e\_TIME e STRING (ver Cap. 5).
- Organization Blocks: contêm todas as SFCs do S7-300/400.
- PID Control Blocks: contêm blocos de função para controle PID.
- S5-S7 Converting Blocks: contêm os blocos padrão que são necessários na conversão dos programas S5 para S7.
- sistema Function Blocks: contêm todas as SFCs do S7-300/400.
- TI-S7 Converting Blocks: contêm as funções padrão geralmente utilizadas p.ex. escalonamento de valores analógicos, etc.

### Notas

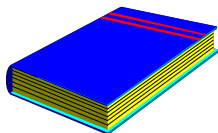
Atentar para o fato que, bibliotecas adicionais são criadas durante a instalação dos pacotes de opção.

Uma descrição das bibliotecas S7 PID e Blocos de Conversão TI - S7 estão localizadas sobre: *Taskbar -> SIMATIC -> S7 manuals -> PID Control, padrão Functions 2*.

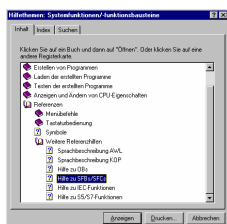
## Fatos interessantes sobre Funções do Sistema (SFC)



As funções do sistema (SFCs e SFBs) são guardadas no sistema operacional das CPU's



sistema Software Reference Manual para S7-300/400 com funções do sistema e funções padrão



Ajuda Online extensiva disponível no software STEP 7

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.4



Conhecimento em Automação  
Training Center

### Introdução

A funcionalidade, que não pode ser implementada com instruções STEP 7 (p.ex. criação de DB, comunicação com outros PLCs, etc.) pode ser implementada no STEP7 com a ajuda das funções do sistema (SFCs) ou blocos de funções do sistema (SFBs).

SFCs e SFBs são blocos que estão guardados no sistema operacional das CPU's em vez da memória do usuário. Por esta razão, a parte das instruções atuais não são transmitidas mas somente a parte das declarações dos SFC's ou SFB's durante a execução da leitura de um SFC ou SFB da CPU.

Com a ajuda do Editor STL/LAD/FBD, a execução da leitura "block" pode ser aberta e a parte declaração mostrada. Uma transmissão no sentido inverso dos SFCs e SFBs para dentro da CPU, deste modo, não é possível.

No programa do usuário, os SFBs e SFCs podem deste modo serem chamados como se fossem FBs ou FCs através da instrução CALL. Com SFBs, um DB de usuário deve ser especificado como DB instance do SFB por esta razão.

Quais SFBs e SFCs estão disponíveis irá depender individualmente do sistema de PLC usado (S7-300 ou S7-400) e da CPU instalada. Os blocos têm, deste modo, indiferentemente de onde estão sendo chamados em um S7-300 ou S7-400, os mesmos números, a mesma funcionalidade e a mesma interface de chamada.

### Manual

Uma descrição adicional das funções do sistema podem ser encontradas no manual:

- The sistema Software Reference Manual for S7-300/400, sistema Functions and Standard Functions.

### Ajuda Online

Exite também uma descrição adicional das funções do sistema no software STEP 7. Chamar o menu help no editor de programa e selecionar a opção:

- *Help topics -> Block help -> Help with SFBs/SFCs .*

## Vista geral das Funções do Sistema (SFC) (Parte 1)

Grupo de Funções	Função	Bloco	S7-300	S7-400
Funções de Bloco e Cópia	Mover bloco	SFC 20	X	X
	Ajustar campo	SFC 21	X	X
	Gerar DB	SFC 22	X	X
	Apagar DB	SFC 23	-	X
	Testar DB	SFC 24	-	X
	Comprimir memória	SFC 25	-	X
	Substituir valor no Accu 1	SFC 44	X <sup>1)</sup>	X
Controle de Programa	Interrupção multiproces.	SFC 35	-	X <sup>2)</sup>
	Tempo do ciclo de gatilho	SFC 43	X	X
	Estado Stop	SFC 46	X	X
	Atraso (Espera)	SFC 47	X <sup>1)</sup>	X
Manipulando o Relógio	Ajusta tempo no relógio	SFC 0	X	X
	Lê tempo do relógio	SFC 1	X	X
	Sincroniza o relógio	SFC 48	-	X
Contador de Horas de Operação	Ajusta o contador	SFC 2	X <sup>1)</sup>	X
	Parte e para	SFC 3	X <sup>1)</sup>	X
	Leitura	SFC 4	X <sup>1)</sup>	X
	Ler tempo do sistema	SFC 64	X	X

1) não para a CPU 312IFM

2) somente para as novas CPUs

## SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.5



Conhecimento em Automação  
Training Center

### Funções de Cópia e Funções de Blocos

- SFC 20 copia o conteúdo da uma área memória (fonte) em outra área de memória (destino).
- SFC 21 preenche uma área de memória (campo destino) com o conteúdo de uma área de memória especificada (campo fonte).
- SFC 22 cria um DB sem valores presetados na memória de trabalho.
- SFC 23 apaga um DB na memória de trabalho e possivelmente na memória de carga.
- SFC 24 determina se um DB está presente na memória de trabalho (e o seu tamanho).
- SFC 25 comprime a memória. Quando os blocos são corrigidos, ficam espaços desocupados na memória que são removidos durante a compressão.
- SFC 44 (chamado no OB 122) salva um valor substituto no Acumulador para um módulo de entrada faltante.

### Controle de programa

- SFC 35 gatilha, em multiprocessamento, a sincronização de partida do OB 60 em todas as CPUs.
- SFC 43 reinicializa a monitoração do ciclo de varredura da CPU.
- SFC 46 leva a CPU para o estado Stop.
- SFC 47 implementa tempos de espera no programa do usuário até 32767 µs.

### Manipulando o Relógio

- SFC 0 ajusta um dado e o horário do dia para o relógio de tempo real da CPU.
- SFC 1 lê um dado e horário do dia correntes na CPU.
- SFC 48 sincroniza todos os relógios escravos presentes em um segmento de barramento de comunicação. Na chamada da CPU devem ser atribuídos seus parâmetros como relógio mestre.

### Contador de horas de operação

A CPU possui um contador específico do número de horas de operação com o qual você pode gravar a duração do tempo de operação do equipamento.

- SFC 2 ajusta o contador de horas de operação para um valor especificado.
- SFC 3 parte e para o contador de horas de operação.
- SFC 4 lê o valor corrente de horas de operação e seu estado.
- SFC 64 lê o tempo do sistema da CPU. O tempo do sistema é um contador que corre livremente fazendo contagens a cada 10 ms (S7-300) ou 1 ms (S7-400)..

## Vista geral das Funções do Sistema (SFC) (Parte 2)

Grupo de Funções	Funções	Bloco	S7-300	S7-400
Transferência de arquivos de dados	Escrita de parâm. dinâmicos	SFC 55	X	X
	Escrita de parâm. definidos	SFC 56	X	X
	Determinação parâm. módulos	SFC 57	X	X
	Escrita de arquivos de dados	SFC 58	X	X
	Leitura de arquivos de dados	SFC 59	X	X
Interrupção por tempo	Ajuste	SFC 28	X <sup>1)</sup>	X
	Cancelamento	SFC 29	X <sup>1)</sup>	X
	Ativação	SFC 30	X <sup>1)</sup>	X
	Varredura	SFC 31	X <sup>1)</sup>	X
Atraso na interrupção	Partida	SFC 32	X <sup>1)</sup>	X
	Cancelamento	SFC 33	X <sup>1)</sup>	X
	Varredura	SFC 34	X <sup>1)</sup>	X
Erros Síncronos	Mascaramento de erros	SFC 36	X	X
	Desmascaramento de erros	SFC 37	X	X
	Leitura registrador de estado	SFC 38	X	X
Erros de interrupção e erros assíncronos	Cancela nova interrupção	SFC 39	X	X
	Habilita nova interrupção	SFC 40	X	X
	Atrasa nova interrupção	SFC 41	X	X
	Habilita interrup. alta prioridade	SFC 42	X	X

1) não para CPU 312IFM

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.6



Conhecimento em Automação  
Training Center

#### Transferência de Dados Arquivados

Existe uma área de dados do sistema com dados de parâmetros e diagnósticos para os parâmetros atribuídos aos módulos. Esta área contém arquivos de dados de 0 a 255 que podem ser lidos ou escritos.

- SFC 55 transfere os parâmetros dinâmicos para o módulo endereçado. Os parâmetros no SDB não são sobreescritos na CPU.
- SFC 56 transfere os parâmetros (dado arquivado RECNUM) para o módulo.
- SFC 57 transfere todos os dados arquivados no SDB para o módulo.
- SFC 58 transfere o arquivo de dados arquivados para o módulo.
- SFC 59 lê o arquivo de dados arquivados do módulo.

#### Interrupções de Tempo

Os blocos são utilizados para processamento controlado do horário do dia (OB 10 a 17). Você pode determinar cada ponto de partida com o software STEP 7 ou com as seguintes funções do sistema.

- SFC 28 ajusta os dados horário do dia de partida de um OB de horário do dia.
- SFC 29 apaga os dados de partida e horário do dia de um OB (OB 10 a OB 17).
- SFC 30 ativa o horário especificado do OB de interrupção.
- SFC 31 verifica o estado de um OB de interrupção de tempo.

#### Interrupção de Atraso

- SFC 32 parte em um modo atraso uma interrupção de atraso (OB 20 to 27).
- SFC 33 cancela uma interrupção de atraso.
- SFC 34 verifica o estado de uma interrupção de atraso.

#### Erros Síncronos

- SFC 36 mascara um erro síncrono, ou seja uma instrução de falha não conduz a chamada de um OB de erro síncrono.
- SFC 37 demascara o erro síncrono
- SFC 38 lê o registrador de erros.

#### Interrupção e Erros Assíncronos

- SFC 39 desabilita o processamento da interrupção e eventos de erros assíncronos.
- SFC 40 habilita novamente o processamento da interrupção e erros assíncronos.
- SFC 41 atrasa o processamento da interrupção e erros assíncronos.
- SFC 42 habilita novamente o processamento da interrupção com atraso e erros assíncronos.



## Vista geral das Funções do Sistema (SFC) (Parte 3)

Grupos de Funções	Funções	Bloco	S7-300	S7-400
Diagnósticos do Sistema	Ler informações de partida.	SFC 6	-	X
	Ler lista parcial estados sistema	SFC 51	X	X
	Escrever buffer de diagnósticos	SFC 52	X	X
Imagem de Processo e área de I/O	Atualizar entradas - PII	SFC 26	-	X
	Atualizar saídas - PIQ	SFC 27	-	X
	Setar um campo bit nas I/Os	SFC 79	-	X
	Resetar um campo bit nas I/Os	SFC 80	-	X
Endereçamento dos módulos	Determinar endereço lógico	SFC 5	-	X
	Determinar um slot	SFC 49	X	X
	Determinar todos end. lógicos	SFC 50	X	X
I/O Distribuído	Gatilhar interrupção hardware	SFC 7	1)	1)
	Sincronizar DP escravos	SFC 11	1)	1)
	Ler diagnóstico de interrupção	SFC 13	1)	1)
	Ler dados do usuário	SFC 14	1)	1)
	Escrever dados do usuário	SFC 15	1)	1)
Comunicação com Dados Globais	Enviar pacote de dados globais	SFC 60	-	X
	Receber pacote dados globais	SFC 61	-	X

1) Somente para CPUs com canal DP, por exemplo CPU 315-2 DP

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.7



Conhecimento em Automação  
Training Center

#### Sistema de Diagnósticos

- SFC 6 lê as informações de partida do último OB chamado e o OB de partida.
- SFC 51 lê uma parte da lista de estados do sistema. A lista contém: dados do sistema, dados de estado de diagnóstico, dados de diagnósticos e o buffer de diagnósticos.
- SFC 52 escreve uma entrada de usuário no buffer de diagnósticos

#### Imagem de Processo e Área de I/O

- SFC 26 atualiza total ou parcialmente a tabela de imagem de processo de entrada.
- SFC 27 transfere total ou parcialmente a imagem de processo para os módulos de saída.
- SFC 79/ 80 são usados para setar e resetar campos binários na área de I/O em conjunto com a função Rele de Controle Mestre.

#### Endereçamento dos Módulos

- SFC 5 fornece o endereço lógico para um endereço geográfico.
- SFC 49 determina o endereço geográfico de um endereço lógico.
- SFC 50 fornece todos os endereços lógicos para um módulo.

#### I/O Distribuído

- SFC 7 gatilha uma interrupção de hardware para o DP mestre. O SFC 7 é chamado no programa do usuário de um escravo inteligente (CPU 315-2DP).
- SFC 11 sincroniza um ou diversos grupos de DP escravos.
- SFC 13 lê os dados de diagnósticos de um DP escravo.
- SFC 14 lê a consistência dos dados de um DP escravo.
- SFC 15 escreve a consistência dos dados de um DP escravo.

#### Comunicação com Dados Globais

- Os dados globais são transferidos ciclicamente (como a cada oitavo ciclo) utilizando o SFC. Com a ajuda dos SFC 60 e 61, envio e recepção de pacotes de dados globais podem ser gatilhados no programa do usuário.
- SFC 60 envia um pacote de dados globais.
  - SFC 61 recebe um pacote de dados globais.



## Vista geral das Funções do Sistema (SFC) (Parte 4)

Grupo de Funções	Funções	Bloco	S7-300	S7-400
Troca de dados utilizando SFB, conexão configurada	Verifica estado	SFC 62	-	X
	Envio não coordenado	SFB 8	-	X
	Recepção não coordenada	SFB 9	-	X
	Bloco de Envio	SFB 12	-	X
	Bloco de Recepção	SFB 13	-	X
	Lê dados de CPU remota	SFB 14	-	X
	Escreve dados de CPU remota	SFB 15	-	X
	Envia para impressora	SFB 16	-	X
	Executa Restart completo	SFB 19	-	X
	Estado Stop	SFB 20	-	X
	Executa Restart	SFB 21	-	X
	Verifica estado de equipamento	SFB 22	-	X
	Recebe estado de equipamento	SFB 23	-	X
Troca de dados utilizando SFC, conexão não configurada	Envia dados externamente	SFC 65	1)	1)
	Recebe dados externamente	SFC 66	1)	1)
	Lê dados externamente	SFC 67	1)	1)
	Escreve dados externamente	SFC 68	1)	1)
	Cancela conexão externamente	SFC 69	1)	1)
	Lê dados internamente	SFC 72	1)	1)
	Escreve dados internamente	SFC 73	1)	1)
	Cancela conexão internamente	SFC 74	1)	1)

1) somente for innovated CPUs

### SIMATIC S7

Siemens AG 1999. All rights reserved.

data: 04.10.2007  
File: PRO2\_07P.8



Conhecimento em Automação  
Training Center

### Troca de dados utilizando SFBs

Os SFBs são utilizados para trocar dados e gerenciar programas utilizando conexões configuradas. Dependendo de quais chamadas de SFB são necessárias para somente um parceiro de comunicação ou para ambos, referência é feita para comunicação de mão única ou de mão dupla. Estes SFBs existem somente no sistema operacional do S7-400.

- SFC 62 determina o estado de um SFB instance local e o estado da conexão associada.
- SFB 8 envia dados para um parceiro remote sem coordenação.
- SFB 9 é o contador do SFB 8.
- SFB 12 envia dados (até 64 KByte) para o parceiro remote com um reconhecimento.
- SFB 13 recebe dados para o parceiro remote com um reconhecimento.
- SFB 14 lê dados de uma CPU remota (comunicação de mão única).
- SFB 15 escreve dados para uma CPU remota (comunicação mão única).
- SFB 16 envia dados com formatação para uma impressora remota.
- SFB 19 gatilha um restart completo para um parceiro remoto.
- SFB 20 transfere o parceiro remote para o estado STOP.
- SFB 21 executa um restart para um parceiro remoto.
- SFB 22 fornece o estado do equipamento (estado de operação, informações de erros) do parceiro remoto.
- SFB 23 recebe o estado do equipamento de um parceiro remoto.

### Troca de dados utilizando SFCs

Esta comunicação – também conhecida como comunicação básica - é implementada com S7- 300 bem como com S7-400. Em comparação com SFB de comunicação as seguintes diferenças aparecem:

- não é necessária configuração de conexão.
- nenhum bloco de dados instance é necessário.
- comprimento máximo de dados de usuário de 76 bytes
- configuração de conexão dinâmica.
- comunicação via MPI ou K bus.

## Vista geral das Funções do Sistema (SFC) (Parte 5)

Grupo de Funções	Funções	Bloco	S7-300	S7-400
Controle Integrado em Malha Fechada	Controle Contínuo	SFB 41	3)	-
	Controle passo	SFB 42	3)	-
	Configuração de Pulso	SFB 43	3)	-
Tecnologia Plástica	Chama bloco assembler	SFC 63	1)	-
Funções Integradas	Contador de alta velocidade	SFB 29	2)	-
	Medidor de Frequência	SFB 30	2)	-
	Contador A/B	SFB 38	3)	-
	Posicionamento	SFB 39	3)	-
Temporizador IEC e Contador IEC	Pulso	SFB 3	X	X
	Atraso na ligação	SFB 4	X	X
	Atraso no desligamento	SFB 5	X	X
	Contador crescente	SFB 0	X	X
	Contador decrescente	SFB 1	X	X
	Contador crescente / decrescente	SFB 2	X	X
Mensagens Referenciadas a Bloco	Mensagem sem reconhecimento	SFB 36	-	X
	Mensagem com reconhecimento	SFB 33	-	X
	Mensagem com 8 valores acompanhados	SFB 35	-	X
	Mensagem sem valores acompanhados	SFB 34	-	X
	Envia arquivo de dados	SFB 37	-	X
	Desabilita arquivo de dados	SFC 10	-	X
	Habilita mensagens	SFC 9	-	X

1) somente para CPU 614

2) somente para CPU 312 IFM

3) somente para CPU 314IFM

## SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.9



Conhecimento em Automação  
Training Center

### Controle Integrado em Malha Fechada

Estes blocos encontram-se integrados nas versões mais recentes de CPUs.

### Tecnologia Plástica

Para a CPU 614 (S7-300), blocos individuais podem ser criados em linguagem "C". A função de sistema SFC 63 é utilizada para chamar blocos.

### Funções Integradas

Estes blocos existem somente para as CPUs 312 IFM (S7-300). Você irá encontrar uma descrição no manual *Integrated Functions*.

- SFB 29 conta pulsos nas entradas integradas da CPU.
- SFB 30 é usada para medir frequências utilizando as entradas integradas.

### Temporizador e Contador IEC

Esta torna disponível temporizadores e contadores que correspondem ao padrão IEC 1131-3. Os temporizadores e contadores remanescentes são implementadas como para SIMATIC S5, por questões de compatibilidade.

Os temporizadores e contadores IEC diferem em uma larga faixa de valores para os valores temporizadores e contadores.

### Mensagens Referenciadas a Blocos

Estes blocos são usados para implementas sistemas de mensagem para sistemas IHM, como para sistemas de controle de processos.

As mensagens são geradas na CPU S7, com este procedimento e as respectivas mensagens incluindo variáveis de processo são enviadas para o equipamento que as mostra identificadamente.

Um conceito de reconhecimento central é usado. Isto é, quando você reconhece uma mensagem no equipamento que a mostra, uma resposta é enviada para a CPU que originou. A informação é distribuída para todos os usuários identificados da CPU.

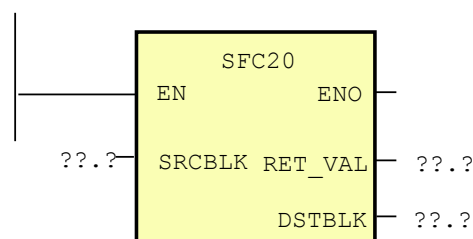
As mensagens são gatilhadas por uma transição do sinal de entrada.

## Chamada das Funções do Sistema (SFC) e Blocos de Funções do Sistema (SFB)

### Funções do Sistema:

```
CALL SFC 20
SRCBLK :=
RET_VAL :=
DSTBLK :=
```

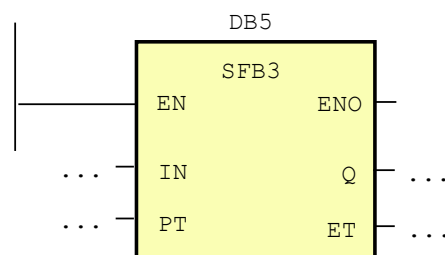
Chamada em STL



Chamada em LAD

### Blocos de Funções do Sistema :

```
CALL SFB 3, DB5
IN :=
PT :=
Q :=
ET :=
```



### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.10



Conhecimento em Automação  
Training Center

### Blocos de Funções do Sistema

Um bloco de funções do sistema SFB é um bloco de funções que estão integrados no sistema operacional de uma CPU S7. Como um resultado, SFBs não são transferidos para dentro de uma CPU como parte do programa do usuário.

Do mesmo modo que os FBs, os SFBs são blocos “com memória”. Eles devem ser atribuídos como instance no programa do usuário.

### Funções do Sistema

Uma função do sistema é uma função que está integrada no sistema operacional da CPU S7. SFCs podem ser chamados do programa do usuário como FCs.

Do mesmo modo que os FCs, os SFCs são blocos “sem uma memória”.

### Chamada

Quando uma função do sistema é chamada, a função do sistema é automaticamente copiada dentro do programa do usuário vigente.

Adicionalmente, todas as funções do sistema são arquivadas na biblioteca padrão Standard Library V3.x, S7-program - SFB. Você pode também copiar os SFCs e SFBs dentro do programa do usuário desta biblioteca.

Uma tabela completa de símbolos (com designações em Inglês) existe na biblioteca. Os símbolos dos blocos utilizados são automaticamente copiados dentro da tabela de símbolos do programa do usuário.

## Avaliação de uma Mensagem de Erro

**A verificação do bit BR (resultado binário) retorna RLO=0 quando ocorrer falha no processamento e RLO=1 quando não ocorrerem falhas.**

- Verificação do BR em STL com “A BR”
- Verificação em LAD utilizando parâmetro de saída ENO

Muitas Funções de Sistema (SFCs) retorna um código de erro com a seguinte configuração no parâmetro de saída RET\_VAL (INT):

- RET\_VAL=W#16#8 X Y Z

**Classe de erro, número individual de erro (específico SFC) ou número do evento (geral)**

**X>0: erro geral, X= No. do parâmetro de falha**

**X=0: erro específico ocorrido com o SFC**

**senal "8": erro ocorrido**

- **Exemplo:**
  - W#16#8081 é um código de erro específico do SFC.
  - W#16#823A é um código de erro geral; o erro foi causado pelo parâmetro No. 2.

SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.11



Conhecimento em Automação  
Training Center

## Informação de Erro

Uma SFC processada mostra a você, no programa do usuário, se a CPU pode executar com sucesso a função SFC ou não. Você recebe a informação do erro correspondente de duas formas:

- no bit BR da palavra de estado e;
- no parâmetro de saída RET VAL (retorno de valor);

## Nota

Você deve sempre proceder da seguinte maneira antes de avaliar o parâmetro de saída específica da SFC :

- antes de tudo avaliar o bit BR da palavra de estado (status word);
- subsequentemente verificar o parâmetro de saída RET VAL;

Se uma falha no processamento da SFC é sinalizada através do bit BR ou um código de erro geral é encontrado no RET\_VAL, você não deve avaliar o parâmetro de saída específico da SFC.

## Erros Gerais

O código de erro geral indica erros que podem ocorrer com qualquer função do sistema. Um código de erro geral consiste de dois seguintes números:

- um número de parâmetro entre 1 e 127, dos quais o 1 indica o primeiro parâmetro, 2 o segundo parâmetro etc., da SFC chamada.
- um número de evento entre 0 e 127. O número do evento indica um erro síncrono.

Uma descrição completa dos códigos de erros gerais pode ser encontrada no manual: "System Functions and Standard Functions" ou na ajuda Online.

## Erros Específicos

Diversas funções do sistema (SFCs) disponibilizam um valor de retorno que fornece um código de erro específico. Este código de erro indica que um erro que pertence a uma função de sistema específica ocorreu durante o processamento da função.

Uma descrição dos códigos de erros específicos pode ser encontrada na ajuda Online para as funções do sistema.

## Exercício 7.1: Geração de um DB com um atributo "UNLINKED"

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.12



Conhecimento em Automação  
Training Center

**Objetivo do exercício** Você gerar um bloco de dados com o atributo "UNLINKED".

### Tarefa

Devido ao fato da memória de trabalho possuir somente um tamanho limitado (usualmente muito pequena), diversos blocos de dados com vários valores de receitas são armazenados somente na memória de carga para gerenciamento de receita.

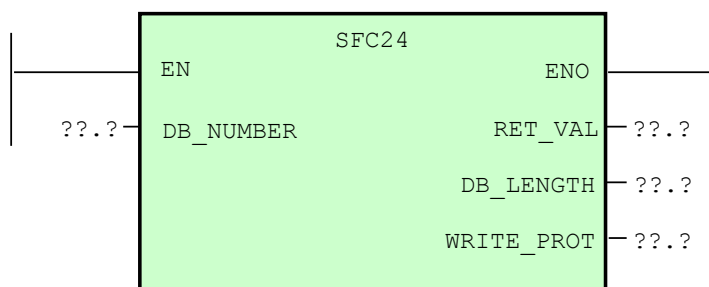
Somente um DB de trabalho, no qual a receita vigente está armazenada, está presente na memória de trabalho. Para uma mudança de receita, os valores requeridos são copiados da memória de carga para a memória de trabalho.

Com a ajuda do atributo "UNLINKED" você se assegura que o bloco de dados estão somente salvos na memória de carga durante a transferência da PG para a CPU e que eles não são automaticamente copiados para dentro da memória de trabalho.

### O que fazer

1. Inserir um DB20.
2. Declarar uma variável "receita" do tipo ARRAY[1..20] como um tipo de componente "INT" no DB20.
3. Com a ajuda do menu de comando *View -> Data View*, altere a visualização para "view" e inicialize os campos individuais com valores em sequência ascendente.
4. Selecione as propriedades do bloco e parametrize o atributo "UNLINKED".
5. Transfira o bloco de dados DB 20 para a CPU.
6. O que acontece quando você, por exemplo, acessa o DB 20 no programa do usuário com a instrução L DB20.DBW0?

## Exercício 7.2: Testando um Bloco de Dados (SFC 24: somente para S7-400)



Parâmetro	Declaração	Tipo dado	Área de Memória	Descrição
DB_número	INPUT	WORD	I, Q, M, D, L, Const.	Número do DB a ser verificado
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Informação de erro
DB_LENGTH	OUTPUT	WORD	I, Q, M, D, L	Número de bytes de dados, que tem no DB selecionado
WRITE_PROT	OUTPUT	BOOL	I, Q, M, D, L	Informação sobre proteção de escrita ID do DB selecionado (1 significa protegido contra escrita)

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.13



Conhecimento em Automação  
Training Center

**Objetivo do Exercício** Com a ajuda da SFC 24 você pode determinar se um bloco de dados específico existe na memória de trabalho ou não.

### Tarefa

Com a ajuda da SFC 24 criar uma FC 72 que determine se existe um DB na memória de trabalho, na memória de carga ou se ele não existe na CPU:

- A FC 72 espera o número do bloco a ser testado no parâmetro de entrada #DB\_NUM (WORD).
- A FC 72 retorna a informação desejada e retorna valor #RET\_VAL (INT) para o bloco chamado:
  - 1: DB existe na memória de carga
  - 0: DB existe na memória de trabalho
  - -1: DB não existe

### Nota

O parâmetro de saída #RET\_VAL do SFC 24 retorna os seguintes identificadores de erros específicos do sistema :

- w#16# 0000 nenhum erro ocorrido
- w#16# 80A1 número incorreto no parâmetro DB\_NUMBER (0 ou > número máx. DB)
- w#16# 80B1 o DB não existe na CPU
- w#16# 80B2 o DB foi gerado com a palavra-chave UNLINKED (é encontrada somente na memória de carga)

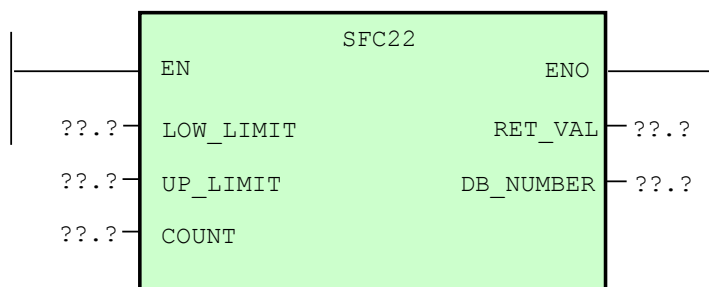
### O que fazer

1. Criar um bloco FC 72
2. Criar um OB1, que com a ajuda do FC 72 verifique se o DB 20 existe ou não. Mostrar a informação retornada no display do Simulador.
3. Transfira os blocos para a CPU e teste seu programa.

### Nota

A função de sistema SFC 24 somente existe para o S7-400!

### Exercício 7.3: Geração de um DB (SFC 22)



Parâmetro	Declaração	Tipo dado	Área de memória	Descrição
LOW_LIMIT	INPUT	WORD	I, Q, M, D, L, Const.	Menor número DB
UP_LIMIT	INPUT	WORD	I, Q, M, D, L, Const.	Maior número DB
COUNT	INPUT	WORD	I, Q, M, D, L, Const.	No. de bytes dados; um número par deve ser especificado aqui
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor do SFC
DB_NUMBER	OUTPUT	WORD	I, Q, M, D, L	Número do DB criado, situa-se entre LOW_LIMIT e UP_LIMIT

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

 dado: 04.10.2007  
 File: PRO2\_07P.14

 Conhecimento em Automação  
 Training Center

**Objetivo do exercício** Você se familiarizar com a criação de um novo DB no programa.

**Tarefa** No OB100 de start-up, um DB 10 será gerado na memória de trabalho. Depois a valores da receita serão copiados para a memória de carga dentro deste DB.

- O que fazer**
1. Criar o OB 100.
  2. Criar o DB 10 com um comprimento de 20 palavras de dados no OB100. Use o SFC 22 para isto (ver acima). Armazene o parâmetro #RET\_VAL na MW 0 e o parâmetro #DB\_NUMBER no display do Simulator.
  3. Transfira o OB 100 para a CPU e teste seu programa.

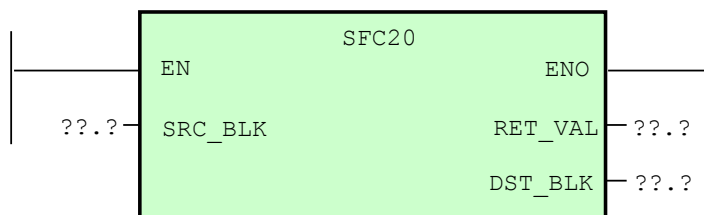
**Nota** Na cópia entre a memória de carga e a memória de trabalho, você deve notar que acessar a memória de carga "lenta" requer consideravelmente mais tempo do que o acesso para a memória de trabalho "rápida".

Se grandes quantidades são copiadas com OB1, o ciclo de tempo, entre outras coisas, deve ser regatilhado.

- Identificadores de erros**
- A função de sistema SFC 22 fornece as seguintes mensagens de erro utilizando o parâmetro #RET\_VAL:
- W#16# 0000 nenhum erro
  - W#16# 8091 limite de tamanho de aninhamento ultrapassado
  - W#16# 8092 compressão de memória está ativa
  - W#16# 80A1 número incorreto de DB
  - W#16# 80A2 tamanho incorreto
  - W#16# 80B1 nenhum número de DB disponível (DB já existe)
  - W#16# 80B2 memória não suficiente
  - W#16# 80B3 memória contínua não suficiente (compressão requerida)



## Exercício 7.4: Copiando um DB da Memória de Carga para a Memória de Trabalho (SFC 20)



Parâmetro	Declaração	Tipo dado	Área memória	Descrição
SRC_BLK	INPUT	ANY	I, Q, M, D, L	Área de memória a ser copiada (= campo fonte). O campo fonte também pode estar presente em um DB pertinente não sequencial na memória de carga (DB, que foi compilado com a palavra-chave UNLINKED)
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor do SFC
DST_BLK	OUTPUT	ANY	I, Q, M, D, L	Área de memória na qual a cópia ocorreu (campo de destino)

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.15



Conhecimento em Automação  
Training Center

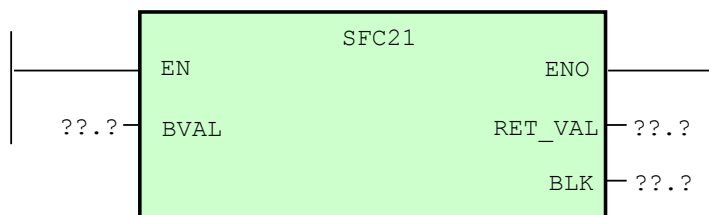
**Objetivo do exercício** Você tornar-se familiar com a função do sistema SFC 20 (BLKMOV).

**Tarefa** Os valores da receita (DW0-DW19) estão sendo copiados do bloco de dados DB 20 para o DB10 (DW0-DW19) na memória de trabalho. A cópia ocorre uma vez após e um impulso na entrada I 0.0 .

- O que fazer**
1. Criar um OB1, que copie os valores da receita do DB 20 para DB 10 com a ajuda do SFC20 (BLKMOV) com um impulso na entrada I 0.0.
  2. Transfere o valor retornado #RET\_VAL para o display digital do simulator.
  3. Transfira seu programa de usuário para a CPU e teste o programa.



## Exercício adicional 7.5: Inicializando um DB (SFC 21)



Parâmetro	Declaração	Tipo dado	Área memória	Descrição
BVAL	INPUT	ANY	I, Q, M, D, L	Preseta o valor
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor do SFC
BLK	OUTPUT	ANY	I, Q, M, D, L	Área de destino, que é inicializada com o conteúdo do BVAL

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.16



Conhecimento em Automação  
Training Center

**Objetivo do exercício:** Tornar-se familiar com o uso de funções do sistema.

### Tarefa

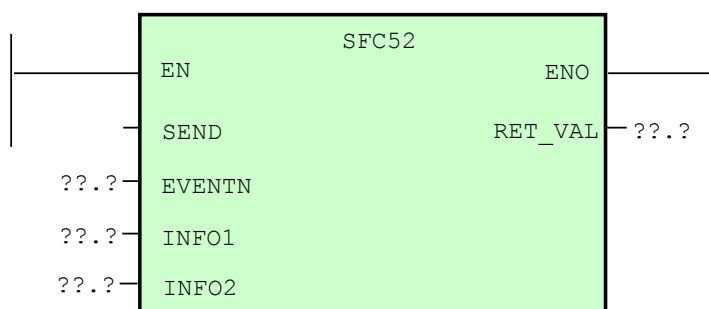
Criar um FC 75 com o qual o bloco de dados pode ser inicializado. O FC 75 tem a seguinte funcionalidade:

- O FC 75 espera os seguintes parâmetros de entrada :
  - #DB\_NUM (WORD): número do DB a ser inicializado
  - #INI (BYTE): Byte amostra com os quais todas as células de memória do DB serão preenchidas.
- O FC 75 antes de tudo determina se o DB desejado existe na memória de trabalho. Se ele existe, então seu comprimento também é determinado.  
Subseqüentemente o FC 75 inicializa o bloco com o byte passado.
- O FC 75 sinaliza no seu #RET\_VAL (BOOL):
  - TRUE: DB foi inicializado com sucesso.
  - FALSE: DB não foi inicializado, isto é, DB não existe na memória de trabalho.

### O que fazer

1. Criar o FC 75.
2. Integrar o FC 75 no OB1 no qual o DB 10 é inicializado com "0" com um impulso na entrada I 1.1.
3. Transfira seu programa para a CPU e teste seu programa.

## Exercício adicional 7.6: Escrevendo uma mensagem no Buffer de Diagnóstico (SFC 52)



Parâmetro	Declaração	Tipo dado	Área memória	Descrição
SEND	INPUT	BOOL	I, Q, M, D, L, Const.	Envio de mensagem para todos os nós logados
EVENTN	INPUT	WORD	I, Q, M, D, L, Const.	Número ou tipo de evento (ID Evento)
INFO1	INPUT	ANY	I, Q, M, D, L	Informação adicional compr. 1 palavra
INFO2	INPUT	ANY	I, Q, M, D, L	Informação adicional compr. 2 palavras
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.17



Conhecimento em Automação  
Training Center

**Objetivo do exercício** Para estar apto a registrar mensagem de programa no buffer de diagnósticos.

#### Tarefa

Criar uma FC 76 com as seguintes funcionalidades:

- Em um erro de sistema (simulado através de um impulso na I1.2), uma mensagem é inserida no buffer de diagnósticos. As mensagens de diagnósticos são adicionalmente postas na PG.

#### O que fazer

- Criar uma FC 76 que insira uma mensagem no buffer de diagnóstico quando existe um "distúrbio no sistema" (impulso em I1.2).
- Ativar a função "CPU Messages" no SIMATIC Manager.
- Chamar a FC 76 no OB1 e testar seu programa.

#### Nota

Usar os seguintes parâmetros para a SFC 52:

- EVENTN W#16# 9B0A (estado contraditório, evento de chegada, erro externo, registro de buffer de diagnóstico)
- INFO1 W#16# 8 (p.ex. número da chave de posição)
- INFO2 DW#16# 1 (p.ex. tipo da chave de posição)

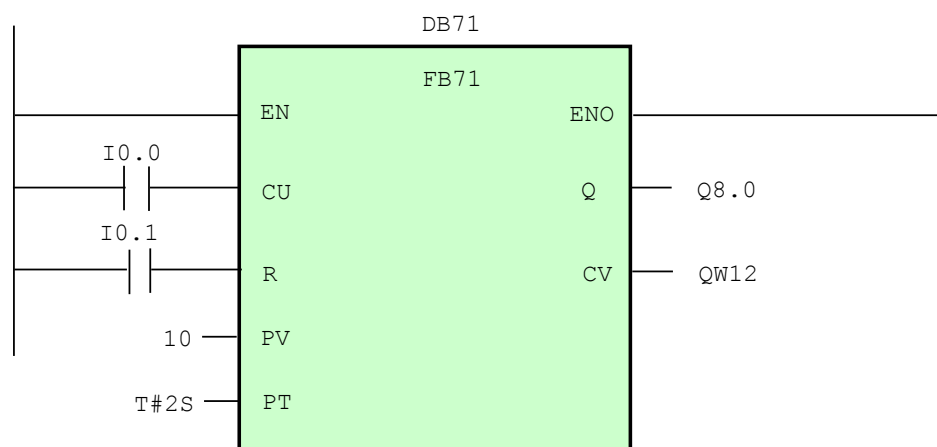
O evento ID número 9 está disponível para o usuário (ver System Functions and Standard Functions Manual).

#### Código de erro

As seguintes mensagens de erro são sinalizadas através do #RET\_VAL da SFC 52 :

- 8083 Tipo de dado INFO1 não permitido
- 8084 Tipo de dado INFO2 não permitido
- 8085 EVENTN não permitido
- 8086 Comprimento do INFO1 não permitido
- 8087 Comprimento do INFO2 não permitido
- 8091 Nenhum nó logado
- 8092 Envio corrente não possível (buffer de envio cheio)

## Exercício adicional 7.7: Bloco Contador com função "Debouncing de Contato"



### SIMATIC S7

Siemens AG 1999. All rights reserved.

data: 04.10.2007  
File: PRO2\_07P.18



Conhecimento em Automação  
Training Center

### Tarefa

Criar um bloco contador de 16 bits (contador crescente) FB71 "CU" com as seguintes propriedades:

- O contador é incrementado de 1 com uma transição positiva, quando o nível de sinal na entrada CU está em 1 pela duração de tempo PT.
- Por outro lado, o bloco contador tem as mesmas características que o contador conforme IEC SFB 0 "CTU".
- A saída Q indica se o valor de contagem corrente é maior do que ou igual ao valor presetado PV.

### Parâmetros

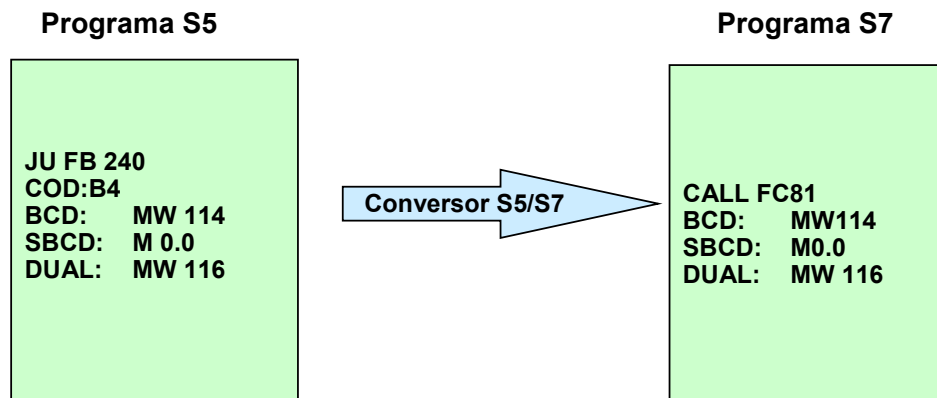
Parâmetro	Declaração	Tipo dado	Descrição
CU	INPUT	BOOL	Entrada de contagem ( <u>C</u> ount <u>u</u> p)
R	INPUT	BOOL	Entrada <u>R</u> eset dominante.
PV	INPUT	INT	<u>P</u> reset <u>V</u> alue (valor presetado).
PT	INPUT	TIME	Periodo de tempo, o nível de sinal tem estar no estado 1 depois de uma transição positiva, só então o contador é incrementado de 1.
Q	OUTPUT	BOOL	Status do contador: Q tem o valor: 1, se CU > PV 0, se contrário
CV	OUTPUT	INT	<u>V</u> alor <u>C</u> orrente

### O que fazer

- Criar uma FB71 com as propriedades desejadas. Usar o bloco de função do sistema SFB0 e SFB4 para implementação.
- Chamar o bloco de contagem FB71 com o instance DB71 no OB1. Atribua os parâmetros de bloco com os seguintes parâmetros atuais:
 

- CU = I0.0	- R = I0.1
- PV = IW4	- PT = T#1000MS
- Q = Q8.0	- CV = QW12 (display digital no simulator)
- Transfira os blocos para a CPU e teste o programa.

## A Biblioteca: Conversão de Blocos S5-S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

data: 04.10.2007  
File: PRO2\_07P.19



Conhecimento em Automação  
Training Center

### Introdução

Esta biblioteca contém blocos padrões S7 necessários para a conversão dos programas S5. Isto significa, se um bloco padrão FB 240, por exemplo, esteve presente no programa S5, o bloco FC 81 na biblioteca substitui o bloco padrão FB 240.

Pelo fato do conversor somente transmitir o bloco FC 81 chamado, você deve copiar o bloco chamado da biblioteca para o seu programa S7.

### Conteúdo da biblioteca

Os blocos da biblioteca são divididos nas seguintes funções:

- Aritmética de ponto flutuante, como adição e subtração
- Funções de sinal, como sinal de dupla frequência de pisca rápido
- Funções integradas, como os conversores de código BCD --> Dual
- Funções lógicas básicas, como LIFO

### Manual

Os blocos são descritos em amplos detalhes no manual "Converting from STEP 5 Programs".

### Ajuda Online

No Editor de Programas, você chama Help --> Help topics --> References --> additional reference aids --> Help with S5/S7 functions.

### Nota

Os flags (memória) chamados de rascunho são também usados para estes blocos, como foi típico para o SIMATIC S5.

## A Biblioteca: Conversão de Blocos TI-S7 (Parte 1)

Bloco	Símbolo	Descrição
FC 80	TONR	Temporizador com atraso na ligação retentivo
FC 81	IBLKMOV	Transfere indiretamente áreas de dados
FC 82	RSET	Reseta uma área de memória bit ou área de I/O
FC 83	SET	Seta uma área de memória bit ou área de I/O
FC 84	ATT	Insere um valor na tabela
FC 85	FIFO	Retira o primeiro valor da tabela
FC 86	TBL_FIND	Procura por um valor na tabela
FC 87	LIFO	Retira o último valor da tabela
FC 88	TBL	Executa a operação tabela
FC 89	TBL_WRD	Copia valor da tabela
FC 90	WSR	Salva o dado no registrador de deslocamento
FC 91	WRD_TBL	Combina logicamente valor com elemento tabela e o salva
FC 92	SHRB	Desloca o bit para o registrador de deslocamento
FC 93	SEG	Gera um modelo de bit para o display digital
FC 94	ATH	Converte caracteres ASCII em um número hexadecimal
FC 95	HTA	Converte um número hexadecimal em caracteres ASCII
FC 96	ENCO	Seta um bit especificado na palavra
FC 97	DECO	Lê o número do bit do bit mais significativo
FC 98	BCDCPL	Gera o complemento de dez
FC 99	BITSUM	Conta o número de bits setados

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.20



Conhecimento em Automação  
Training Center

#### FC 80

A função FC80 parte o temporizador como atraso na ligação com memória (TONR). A FC80 acumula o valor de tempo até que o valor de tempo corrente do tempo executado (#ET) seja o mesmo que o valor presetado (#PV) ou ultrapasse-o.

#### FC 81

Função indireta de transferência de faixas de dados, (IBLKMOV), você pode transferir uma faixa de dados consistindo de bytes, palavras, inteiros (16 bits), palavras duplas, ou duplo inteiros (32 bit) de uma fonte para um destino. Os ponteiros #S\_DATA e o #D\_DATA, tem por tarefa, estruturar o tipo de dado "POINTER" que determina o início da área fonte e a área destino. O comprimento da área a ser copiado está determinado através de parâmetros separados.

#### FC 82/83

Ajusta o estado dos bits em uma área especificada para "1" (FC 83) ou para "0" (FC 82), se o bit MCR é "1." Se o bit MCR é "0", o estado do sinal dos bits na área não é alterado.

#### FC 84-FC92

Estes tratam com funções de tabela para implementar funções FIFO, por exemplo. Os valores são para serem inseridos no formato de palavras e o comprimento é ajustável.

#### FC 93-FC 99

Este grupo torna disponíveis diversas funções de conversão.

## A Biblioteca: Conversão de Blocos TI-S7 (Parte 2)

Bloco	Símbolo	Descrição
FC 100	RSETI	Reseta uma área de saída imediatamente
FC 101	SETI	Seta uma área de saída imediatamente
FC 102	DEV	Desvio padrão
FC 103	CDT	Tabela de dados correlacionados
FC 104	TBL_TBL	Tabela de operações lógicas
FC 105	SCALE	Escalonamento de valor
FC 106	UNSCALE	Desescalonamento de valor
FB 80	LEAD_LAG	Algoritmo Lead/Lag
FB 81	DCAT	Interrupção de controle discreto
FB 82	MCAT	Interrupção de controle Motor
FB 83	IMC	Comaração de matriz índice
FB 84	SMC	Varredura de matriz
FB 85	DRUM	DRUM (processador de seqüência)
FB 86	PACK	Tabela de dados coletados/distribuídos

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.21



Conhecimento em Automação  
Training Center

#### FC 100-FC 101

A função (RSETI) reseta o estado do sinal de bits em uma faixa especificada de bytes para "0" ou para "1" pelo FC 101, se o bit MCR for "1." Se o bit MCR for "0", o estado do sinal dos bytes na faixa não for alterada.

#### FC 102

A função desvio padrão (DEV) calcula o desvio padrão de um grupo de valores armazenados na tabela (TBL). O resultado é armazenado em OUT. O desvio padrão é calculado de acordo com a seguinte fórmula :

$$\text{Desvio padrão} = \sqrt{\frac{(N \times \text{SqSum}) - \text{Sum}^2}{N \times (N - 1)}}$$

Com:

- #Sum = Soma dos valores na TBL N = número de valores na TBL
- #SqSum = Soma de todos os valores na TBL ao quadrado

#### FC 103

A função "tabela de dados correlatos" (CDT) compara um valor de entrada (#IN) com uma tabela já existente de valores de entrada (#IN\_TBL) e procure pelo primeiro valor que seja maior ou igual ou igual ao valor de entrada com a ajuda do índice do valor locado, o valor é então copiado para o respectivo valor de saída (#OUT) na tabela de valores de saída (#OUT\_TBL).

#### FC 104-FC 105

É usado para escalonar valores analógicos de uma entrada analógica ou para uma saída analógica.

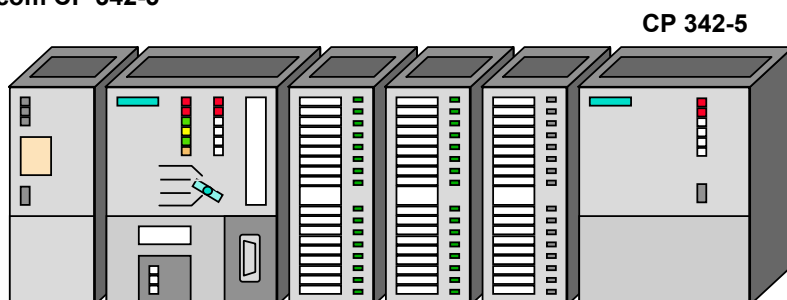
#### FB 80- FB 86

Referem-se ao manual eletrônico.

## A Biblioteca: Blocos de Comunicação

Bloco	Símbolo	Descrição
FC 1	DP_SEND	Envia dado para PROFIBUS-CP
FC 2	DP_RECV	Recebe dados do PROFIBUS-CP
FC 3	DP_DIAG	Dado de diagnóstico de carga de estação
FC 4	DP_CTRL	Tarefa de controle de envio para CP

**Exclusivamente na configuração:  
CPU S7-300 com CP 342-5**



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
File: PRO2\_07P.22



Conhecimento em Automação  
Training Center

### Vista Geral

As funções da biblioteca FC1, FC2, FC3 e FC4 são usadas exclusivamente na seguinte configuração:

- CPU S7-300 com CP 342-5 PROFIBUS externa

Em todos os outros casos, isto é, com S7-300 com interface PROFIBUS-DP integrada e com o sistema S7-400 total, a respectiva funcionalidade é implementada utilizando a carga padrão e transfere comandos (L ... , T...) ou utilizando SFC14 (DPRD\_DAT), SFC15 (DPWR\_DAT), SFC11 (DPSYC\_FR) e SFC13 (DPNRM\_DG).

### FC1

O bloco DP\_SEND passa o dado de uma especificada área de saída DP para o PROFIBUS-CP pela passagem ao I/O distribuído.

### FC2

O bloco DP\_RECV adota o dado de processo do I/O distribuído bem como uma informação de estado em uma especificada faixa DP de entrada.

### FC3

O bloco FC DP\_DIAG é usado para requisição de informação de diagnóstico. Diferenciação é feita entre os seguintes tipos de tarefas:

- requisição de lista de estações DP;
- requisição de lista DP\_DIAGNOSTIC;
- requisição de diagnósticos DP simples;
- leitura de dados de entrada / saída de um escravo DP acíclico;
- leitura do modo de operação DP.

### FC4

O bloco FC DP\_CTR passa tarefas de controle para o PROFIBUS-CP. Diferenciação é feita entre os seguintes tipos de tarefas:

- Controle Global acíclico / cíclico;
- Apaga diagnósticos velhos;
- Seta corrente modo de operação DP;
- Seta modo de operação DP para stop PLC/CP;
- Leitura de dados de entrada / saída ciclicamente;
- Seta o modo de processamento do escravo DP.

## A Biblioteca: Blocos de Controle PID

Bloco	Símbolo	Descrição
FB 41	CONT_C	Bloco de controle PID contínuo
FB 42	CONT_S	Bloco de controle PI em passos
FB 43	PULSEGEN	Bloco gerador de pulso

### SIMATIC S7

Siemens AG 1999. All rights reserved.

dado: 04.10.2007  
 File: PRO2\_07P.23



Conhecimento em Automação  
 Training Center

#### FB 41

O SFB "CONT\_C" (controlador contínuo) é usado nos controladores lógicos programáveis SIMATIC S7 para controle de processos técnicos com variáveis de entrada e saída contínuas. Durante a atribuição dos parâmetros, você pode ativar ou desativar subfunções do controlador PID para adaptar o controlador ao processo.

Você pode utilizar o controlador como um PID como controlador com setpoint fixo ou controle multi-loop como uma cascata, misturador ou controlador proporcional. As funções do controlador são baseados no algoritmo de controle PID com um sinal de saída analógica, se necessário estendido pela inclusão de um estágio gerador de pulso para gerar sinais de saída com largura modulada para dois ou três estágios controladores para atuadores proporcionais.

#### FB42

O SFB "CONT\_S" (controlador de passo) é utilizado nos controladores lógicos programáveis SIMATIC S7 para controle de processos técnicos com valor de sinal de saída digital manipulada para atuadores integrais. Durante a atribuição dos parâmetros, você pode ativar ou desativar subfunções do controlador PI de passos para adaptar o controlador ao processo.

Você pode utilizar o controlador como um controlador PI de setpoint fixo ou em malha de controle em cascata, misturador ou controlador proporcional. As funções do controlador são baseadas no algoritmo suplementadas pelas funções para geração de sinais de saída binária de sinal de atuação analógico.

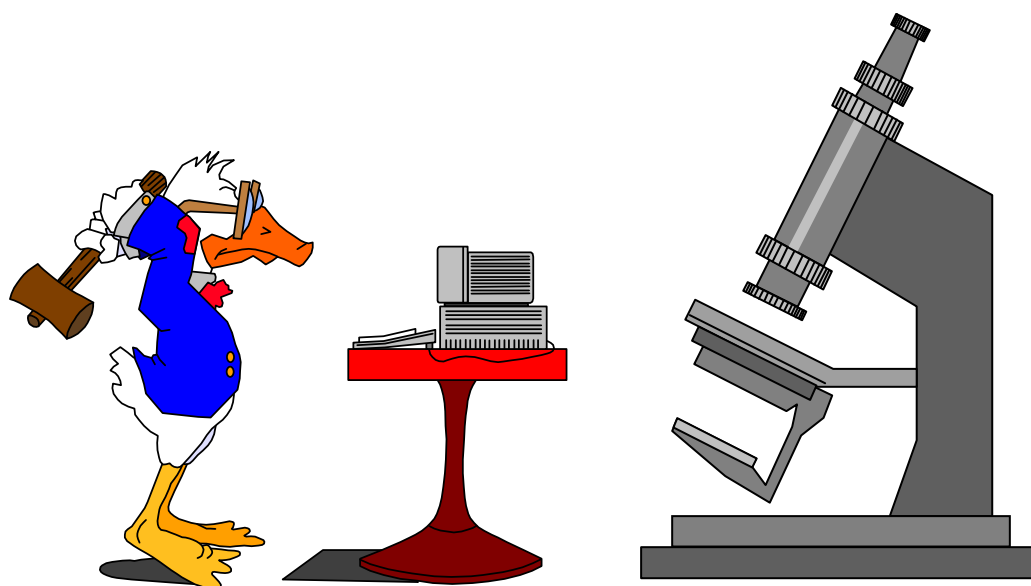
#### FB43

O SFB43 "PULSEGEN" (gerador de pulsos) é usado para estruturar um controlador PID com saída de pulsos para atuadores proporcionais.

Utilizando o SFB "PULSEGEN", controladores PID com dois ou três passos com modulação de largura de pulso pode be configurado. A função é normalmente usada em conjunto com o controlador contínuo "CONT\_C".



## Lidando com Erros Síncronos e Assíncronos



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.1Conhecimento em Automação  
Training Center

### Conteúdo

Pág.

Manipulando Erros Assíncronos .....	2
Manipulando os Blocos de Organização de Erros .....	3
Exemplo de um OB de Erro Assíncrono .....	4
Manipulando Erros Síncronos .....	5
Informações de partida do OB121 para Erros de Programação .....	6
Informações de partida do OB122 para Erros de Acesso .....	7
Máscara de Erros Síncronos .....	8
SFC 36 para Máscara de Falhas Síncronas .....	9
Estrutura de programação de Filtro de Falhas .....	10
Estrutura de acesso ao Filtro de Falhas .....	11
SFC 37 para desmascaramento de Falhas Síncronas .....	12
SFC 38 para leitura do registrador de Erros .....	13
Exemplo: Testando Bloco de Dados .....	14
Exercício 8.1: Manipulação de Erro no FC81 .....	15

## Manipulando Erros Assíncronos

**Erros Assíncronos não são atribuídos a posição particular do programa, isto é, eles aparecem assíncronos para o processamento do programa.**

Tipo de Erro	Exemplo	OB de erro
Erro de Tempo	Máx. tempo de varredura excedido	OB 80
Erro da Fonte de Alimentação	Falha da bateria de backup	OB 81 <sup>2)</sup>
Diagnóstico de Interrupção	Quebra-de-fio na entrada de um módulo com capacidade de interrupção	OB 82
Removendo/Inserindo módulo de Interrupção	Removendo um módulo de sinal no S7-400 durante o modo de operação (RUN)	OB 83 <sup>1)</sup>
Erro de Hardware CPU	Nível de sinal de falha na interface MPI	OB 84 <sup>1)</sup>
Erro de sequência no programa	Erro na atualização da imagem de processo (módulo com defeito)	OB 85
Defeito no bastidor ou bastidor	Defeito em fonte de alimentação de bastidor de expansão	OB 86 <sup>1)</sup>
Erro de comunicação	Identificador de mensagem incorreta	OB 87

<sup>1)</sup> somente com S7-400

<sup>2)</sup> não vai p/Stop sem OB de erro

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.2



Conhecimento em Automação  
Training Center

#### Introdução

O slide acima relaciona os eventos de erros assíncronos. Estes erros não são atribuídos para determinadas posições do programa.

#### Erro de Tempo

A monitoração do tempo de ciclo de varredura tem um ajuste padrão de 150ms. O sistema reconhece um erro de tempo se a duração do ciclo é maior do que 150 ms. Se o erro ocorre duas vezes no mesmo ciclo, a CPU vai para o estado de Stop.

#### Erro na Fonte de Alimentação

Ocorre com a falha ou a perda da bateria de backup e adicionalmente para o S7-400 com o defeito da alimentação de 24 V no bastidor central ou bastidor de expansão.

Diferentemente de outros tipos de erros, sem um OB de erro existente, a CPU mantém-se em estado de Run e um led vermelho de erro acende na CPU.

#### Diagnóstico de Interrupção

Diagnóstico capacita módulos, como por exemplo módulos analógicos, a poderem gatilhar um diagnóstico de interrupção no caso de um erro. Aos módulos devem ser atribuídos parâmetros para tanto. Neste caso o diagnóstico de interrupção é habilitado.

#### Interrupção de Remoção/Inserção

É gatilhado pela inserção ou remoção de módulos no sistema de PLC S7-400. Na inserção de módulos, o sistema operacional verifica se o tipo de módulo correto foi inserido. Esta função permite a remoção e inserção de módulos durante o ciclo de programa.

#### Erro de CPU-H/W

No S7-400, erros são reconhecidos na interface MPI através do K-Bus ou no módulo de interface para I/O distribuído.

#### Erro de sequência de programa

Resulta de erros de acesso à periferia (I/O) na atualização da imagem de processo ou por exemplo, da perda de OB para interrupção parametrizada horário do dia (time-of-day).

#### Defeito em Bastidor

É reconhecido quando um bastidor, uma subrede em sistemas de PLC em rede ou uma estação de periferia (I/O) distribuída falhar.

#### Erro de Comunicação

Um identificador de mensagem incorreto no recebimento de dados globais está presente no S7-300 ou o bloco de dados é muito curto para armazenamento do estado (status) da informação. No S7-400 existem diversas causas, por exemplo, o envio de sincronização de mensagens não é possível.

## Manipulando os Blocos de Organização de Erros

- De forma a impedir que a CPU vá para Stop no caso de um erro, transferir um OB de erro vazio.
- Você pode programar a resposta desejada no OB de erro e, se necessário, requisitar o estado Stop com a função do sistema SFC 46 depois da execução do OB de erro.
- Um identificador de erro adicional é armazenado nas informações de partida do OB de erro, o qual pode se avaliado no programa.
- Uma descrição dos OB's de erro podem ser encontrados no "ajuda" Online ou do sistema e Manual de Funções Padrão.
- A transmissão dos OBs de erro que não são suportados por determinadas CPU são rejeitados com uma mensagem de erro.

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.3Conhecimento em Automação  
Training Center

### Informação de partida

Favor prestar atenção às regras de uso dos OB's de erro.

Para cada bloco de organização, variáveis temporárias são definidas na parte de declaração. O sistema operacional armazena as informações de partida nestas variáveis. O sistema operacional armazena informações adicionais nas informações de partida quando o bloco é chamado.

Como um exemplo você pode ver as informações de partida no OB 81.

Name	Type	Initial Value	Comment
OB81_EV_CLASS	BYTE		16#39, Event class 3, Entering ev
OB81_FLT_ID	BYTE		16#XX, Fault identification code
OB81_PRIORITY	BYTE		26/28 (Priority of 1 is lowest)
OB81_OB_NUMBR	BYTE		81 (Organization block 81, OB81)
OB81_RESERVED_1	BYTE		Reserved for system
OB81_RESERVED_2	BYTE		Reserved for system
OB81_MDL_ADDR	INT		Reserved for system
OB81_RESERVED_3	BYTE		Reserved for system
OB81_RESERVED_4	BYTE		Reserved for system
OB81_RESERVED_5	BYTE		Reserved for system
OB81_RESERVED_6	BYTE		Reserved for system
OB81_DATE_TIME	DATE_AND_TIME		Date and time OB81 started

A variável OB81\_FLT\_ID tem o seguinte significado:

- B#16#21: Pelo menos uma bateria de backup do bastidor central está esgotada (BATTF)
- B#16#22: Tensão de backup no bastidor central está perdida (BAF).
- B#16#23: Defeito da alimentação 24V no bastidor central / eliminado.
- B#16#31: Pelo menos uma bateria de backup de um bastidor de expansão está esgotada.
- B#16#32: Tensão de backup em um dos bastidores de expansão está esgotada.
- B#16#33: defeito da alimentação 24V em um bastidor de expansão.

## Exemplo de um OB de Erro Assíncrono

OB81: OB de erro: defeito na fonte de alimentação

Network 1: defeito na bateria, chegada do evento

```

L   #OB81_FLT_ID           // Carrega identificador de erro
L   B#16#22                 // Identificador: defeito bateria no
==I                                // bastidor central (CR)
=   M   81.1                // Seta flag de memória auxiliar
L   #OB81_EV_classe        // Identificador: entrando, saindo
L   B#16#39                 // Identificador: entrando evento
==I
=   M   81.2                // Flag mem. aux. entrando evento
A   M   81.1                // Defeito na bateria e
A   M   81.2                // entrando evento
S   M   81.0                // Seta flag mem. aux. para mostrar
                                // erro

```

Network 2: Reseta flag de memória auxiliar, quando bateria O.K.

```

L   #OB81_EV_classe        // Identificador: entrando, saindo
L   B#16#38                 // Identificador: saindo
==I
R   M   81.0                // Reset flag de memória auxiliar

```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.4



Conhecimento em Automação  
Training Center

### Tarefa

O defeito na bateria deve resultar na amostragem de um erro na console operacional. Depois de trocada a bateria a mensagem deve automaticamente desaparecer.

### Descrição

Em erros de fonte de alimentação p.ex. falha de bateria, o OB de erro é chamado uma vez pelo sistema operacional. Depois do erro ser eliminado o OB 81 é chamado mais uma vez.

No programa exemplo, a variável #OB81\_FLT\_ID é avaliada, de forma a determinar se existiu um defeito na bateria. Neste caso a variável contém o valor 22H. A comparação desta forma é preenchida e o bit de memória M 81.1 é gatilhado.

O erro mostrado será inicializado quando a bateria tiver falhado (entrando evento) e limpada depois que o erro tenha sido eliminado (saindo evento).

Os seguintes identificadores estão na variável #OB81\_EV\_classe:

- B#16#39 entrando evento
- B#16#38 saindo evento.

O “set” e o “reset” do flag de memória auxiliar M 81.0 é arquivado através da avaliação destas variáveis.

No programa cíclico, o flag de memória auxiliar M81.0 pode ser lincado à uma memória geradora de pulso e ser atribuída a uma saída. A saída então piscará enquanto a bateria estiver esgotada ou removida.

## Manipulando Erros Síncronos

- Erros síncronos são atribuídos diretamente a uma posição no programa do usuário
- Erros em instruções aritméticas (overflow, REAL número inválido)



**Ajuste de Bits de Status**

**Erros no processamento de instruções STL (erro síncrono)**



**Chamada do OB de erro síncrono**

Tipo de erro	Exemplo	OB de erro
Erro de programação	Bloco chamado não existe na CPU	OB 121
Erro de acesso	Acesso direto a um módulo com defeito ou não existente	OB 122

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.5



Conhecimento em Automação  
Training Center

### Erros síncronos

O sistema operacional da CPU gera uma falha síncrona, quando um erro ocorre em conexão imediata com o programa em processamento.

OB121 é chamado com a programação de um erro. OB122 é chamado com um erro de acesso. Se o OB de erro síncrono não é carregado na CPU, a CPU é chaveada para o modo STOP quando a falha síncrona ocorre.

O OB de erro síncrono tem a mesma prioridade que o bloco no qual o erro ocorreu. Por esta razão, os registradores do bloco interrompido pode ser acessado no OB de erro síncrono e este é o porque o programa no OB de erro síncrono pode também retornar os registradores (se necessário com mudança de conteúdo) para o bloco interrompido.

### Mascaramento de erros síncronos

S7 tem os seguintes SFCs, com as quais você pode mascarar e demascarar os eventos de partida do OB121 enquanto o seu programa está sendo processado:

- SFC36 (MSK\_FLT): mascara o código de erro específico
- SFC37 (DMSK\_FLT): demascara o código de erro que foi mascarado pelo SFC36
- SFC38 (READ\_ERR): lê o registrador de erro

## Informações de partida do OB121 para Erros de Programação

Nome da variável	Tipo de dado	Descrição, parametrização
OB121_EV_classe	BYTE	B#16#25= Chamada erro de programação OB121
OB121_SW_FLT	BYTE	Código de erro (ver texto)
OB121_PRIORITY	BYTE	Classe de prioridade na qual o erro ocorreu
OB121_OB_NUMBER	BYTE	Número do OB (B#16#79)
OB121_BLK_TYPE	BYTE	Tipo de bloco interrompido (somente S7-400) OB: B#16#88, DB: B#16#8A, FB: B#16#8E, FC: B#16#8C
OB121_RESERVED_1	BYTE	Adição ao código de erro (ver texto)
OB121_FLT_REG	WORD	OB121: fonte do erro
OB121_BLK_NUM	WORD	Número de blocos no qual o erro ocorreu
OB121_PRG_ADDR	WORD	Endereço do erro no bloco causador do erro (somente S7-400)
OB121_DATE_TIME	DT	Momento da gravação do erro de programação

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.6Conhecimento em Automação  
Training Center

### Código de erro (#OB121\_SW\_FLT)

B#16#21: erro de conversão BCD. A variável #OB121\_FLT\_REG contém um identificador para o registrador respectivo (W#16#0000: ACCU 1).

B#16#22: Dimensão da faixa de erro durante leitura.

B#16#23: Dimensão da faixa de erro durante escrita.

B#16#28: Acesso indireto leitura de BYTE, WORD ou DWORD com endereço de bit diferente de 0 (combinado durante leitura).

B#16#29: Acesso indireto escrita de BYTE, WORD ou DWORD com endereço de bit diferente de 0 (combinado durante escrita).

Neste caso, #OB121\_FLT\_REG contém o endereço do byte de falha e #OB121\_RESERVED\_1 contém o tipo do acesso e área de memória :

Bit 7 a 4 (tipo de acesso): Bit 3 a 0 (área de memória)

0: acesso Bit	0: área I/O	4: DB Global
1: acesso Byte	1: PII	5: DB Instance
2: acesso Word	2: PIQ	6: dado local próprio
3: acesso Double word	3: Memória Bit	7: dado local do chamado

B#16#24: Faixa de erro durante leitura

B#16#25: Faixa de erro durante escrita

#OB121\_FLT\_REG contém o identificador B#16#86: área dado local próprio.

B#16#26: Erro com N°. de temporizador (N°. inválido no #OB121\_FLT\_REG)

B#16#27: Erro com N°. de contador (N°. inválido no #OB121\_FLT\_REG)

B#16#30: Acesso escr.DB global proteg.contra escr.(No.no#OB121\_FLT\_REG)

B#16#31: Acesso escr.DB inst.proteg.contra escr.(No.no#OB121\_FLT\_REG)

B#16#32: No. erro em acesso to global DB (No. no #OB121\_FLT\_REG)

B#16#33: No. erro em acesso to instance DB (No. no #OB121\_FLT\_REG)

B#16# 34: Número do erro em chamada de FC (No. no #OB121\_FLT\_REG)

B#16#35: Número do erro em chamada de FB (No. no #OB121\_FLT\_REG)

B#16#3A: Acesso a DB não carregado (No. no #OB121\_FLT\_REG)

B#16#3C: Acesso a FC não carregado (No. no #OB121\_FLT\_REG)

B#16#3D: Acesso a SFC não carregado (No. no #OB121\_FLT\_REG)

B#16#3E: Acesso a FB não carregado (No. no #OB121\_FLT\_REG)

B#16#3F: Acesso a SFB não carregado (No. no #OB121\_FLT\_REG)

## Informações de partida do OB122 para Erros de Acesso

Nome da variável	Tipo de dado	Descrição, parametrização
OB122_EV_CLASS	BYTE	B#16#29= Chamada erro acesso
OB122_SW_FLT	BYTE	Código de erro (valores possíveis : B#16#42, B#16#43, B#16#44, B#16#45)
OB122_PRIORITY	BYTE	Classe de prioridade na qual o erro ocorreu
OB122_OB_NUMBR	BYTE	Número do OB (B#16#80)
OB122_BLK_TYPE	BYTE	Tipo de bloco interrompido(somente S7-400) OB: B#16#88, DB: B#16#8A, FB: B#16#8E, FC: B#16#8C
OB122_MEM_AREA	BYTE	Adição ao código de erro (ver texto)
OB122_FLT_REG	WORD	OB122: identificador do endereço onde ocorreu o erro.
OB122_BLK_NUM	WORD	Número do bloco no qual o erro ocorreu
OB122_PRG_ADDR	WORD	Endereço do erro no bloco causador do erro (somente S7-400)
OB122_DATE_TIME	DT	Momento da gravação do erro de programação.

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.7Conhecimento em Automação  
Training Center**Código de erro**

B#16#42

A variável #OB122\_SW\_FLT tem o seguinte significado:

S7-300: Erro acesso I/O, leitura

S7-400: Primeiro acesso leitura depois de que ocorre erro

B#16#43:

S7-300: Erro acesso I/O, escrita

S7-400: Primeiro acesso escrita depois de que ocorre erro

B#16#44:

Somente para S7-400: erro no n-éssimo (n&gt;1) acesso leitura depois que ocorre erro.

B#16#45:

Somente para S7-400: erro no n-éssimo (n&gt;1) acesso escrita depois que ocorre erro.

**OB122\_MEM\_AREA**

A variável #OB122\_MEM\_AREA contém informação sobre o tipo de acesso e a área de memória:

Bit 7 a 4 tipo de acesso:

0: Acesso Bit

1: Acesso Byte

2: Acesso Word

3: Acesso Double word

Bit 3 a 0 área de memória:

0: Área I/O

1: Tabela da Imagem de Processo de Entrada

2: Tabela da Imagem de Processo de Saída



## Máscara de Erros Síncronos

### Desvantagens do OB de erro síncrono :

- Código para o gerenciamento de processo e para a manipulação de erro é distribuído entre pelo menos dois blocos
- Problemas com subseqüentes mudanças ou com manutenção

### Melhor:

- Código para gerenciamento de processo e para a manipulação de erro é no mesmo bloco

### Mascaramento da falha síncrona:

- Antes de instruções "críticas":  
SFC 36 MSK\_FLT: mascara falhas síncronas (OB12x – inibe a chamada)
- Executa instruções "críticas"
- Avalia se um erro ocorreu  
SFC 38 READ\_ERR: lê registrador de erro
- OB12x - habilita a chamada uma vez novamente:  
SFC 37 DMSK\_FLT: demascara falhas síncronas

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.8



Conhecimento em Automação  
Training Center

### Desvantagens dos OBs de erro síncrono

A manipulação dos eventos de erros síncronos pelo significado dos OBs de erros síncronos tem algumas desvantagens:

- Com uma manipulação de erro qualificado, uma correspondente avaliação de erro no OB de erro síncrono deve ser executado para cada bloco com instruções que podem gatilhar um erro síncrono.  
Dentro do OB de erro síncrono um considerável tabalho deve desta forma ser executado de forma a localizar o erro no programa do usuário e então poder reagir de acordo.
- Cada mudança em um bloco existente conduz a mudanças correspondentes no OB de erro síncrono.
- Blocos não podem ser integrados em um programa do usuário sem a correspondente consideração no OB de erro síncrono.

### Alternativas aos OBs de erro síncrono

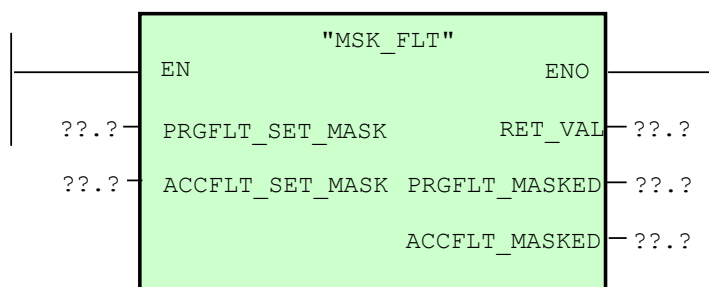
S7 oferece, com the ajuda da função "Máscara de Erros Síncronos", um mecanismo que atribui o código para o gerenciamento de processo e para a manipulação de erro associado a ser instalado no mesmo bloco.

Isto toma lugar, por exemplo, nos seguintes passos:

1. Antes da execução de instruções "críticas" (p.ex. abertura de um DB, ou acesso a um DB de comprimento desconhecido), o correspondente erro síncrono pode ser mascarado pela ação do SFC 36 (MSK\_FLT).  
se an instruction então fails, no síncrono erro OB é chamado.
2. Depois da execução de instruções "críticas", você pode verificar pela ação do SFC 38 (READ\_ERR), enquanto ainda não ocorreram erros críticos e reagir de acordo.
3. Uma vez concluídas as atividades, a falha síncrona previamente mascarada pode então ser desmascarada e então a chamada do OB de erro síncrono é habilitado novamente.



## SFC 36 para Máscara de Falhas Síncronas



Parâmetro	Declaração	Tipo dado	Área de memória	Descrição
PRGFLT_SET_MASK	INPUT	DWORD	I, Q, M, D, L, Const.	Nova (adicional) programação filtro de falhas
ACCFLT_SET_MASK	INPUT	BYTE	I, Q, M, D, L, Const.	Novo (adicional) acesso filtro de falha
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna o valor do SFC, W#16#0001: o novo filtro fica com área em comum com o filtro existente
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Programação completa filtro de falhas
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Acesso completo filtro de falhas

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.9Conhecimento em Automação  
Training Center

### Mascarando erros síncronos

Com o SFC 36 (MSK\_FLT), você inibe a chamada do OB de erro síncrono usando filtros de falhas. Com o nível lógico "1" você identifica nos filtros de falhas para quais erros síncronos o OB não será chamado (as falhas síncronas serão "mascaradas").

O mascaramento especificado é ajustado sobre a máscara armazenada no sistema operacional (operação lógica OU dos bits filtrados). SFC36 sinaliza no valor retornado se, para a máscara especificada nos parâmetros de entrada, a máscara já existe (W#16#0001) para pelo menos um bit.

O SFC36 entrega em sua saída todos os parâmetros atualmente mascarados e indicados com nível lógico "1".

### Reação da CPU

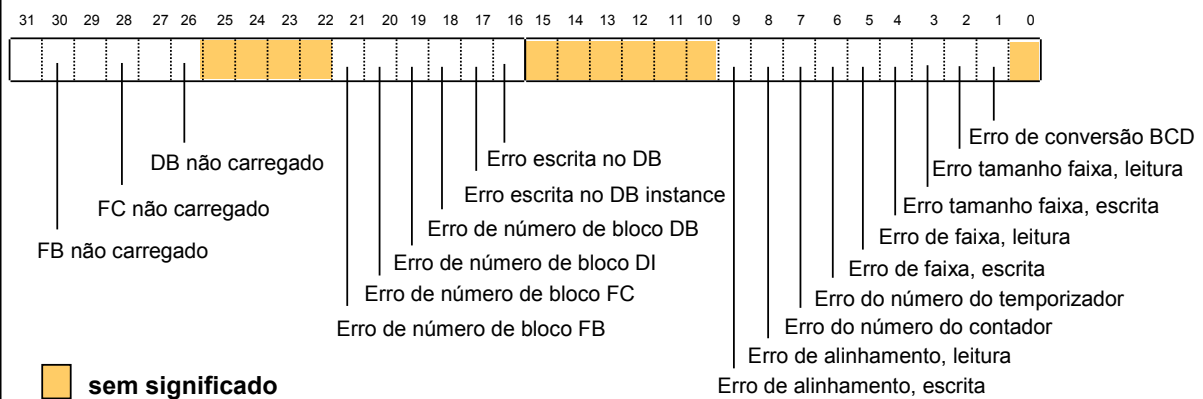
Quando um erro de programação ou acesso é mascarado, então a CPU reage de acordo com o seu tipo de erro:

1. O OB de erro não é chamado para erros de programação ou acesso.
2. O evento de erro é inserido no registrador de erro. O registrador de erro pode ser lido com a ajuda do SFC38 (READ\_ERR).
3. O sistema operacional insere a falha síncrona no buffer de diagnóstico independentemente da máscara.

### Validade do mascaramento

A máscara somente é válida para a classe de prioridade na qual o SFC 36 foi chamado. Se você, por exemplo, inibir a chamada do OB de erro síncrono em um programa principal, o OB de erro síncrono irá continuar a ser chamado se o erro ocorrer em uma interrupção de programa.

## Estrutura de programação de Filtro de Falhas



**Nota:** Os bits correspondentes do parâmetro de saída PRGFLT\_MASKED são setados como a seguir:

Valor = "1": erro está mascarado.

Valor = "0": erro não está mascarado.

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.10



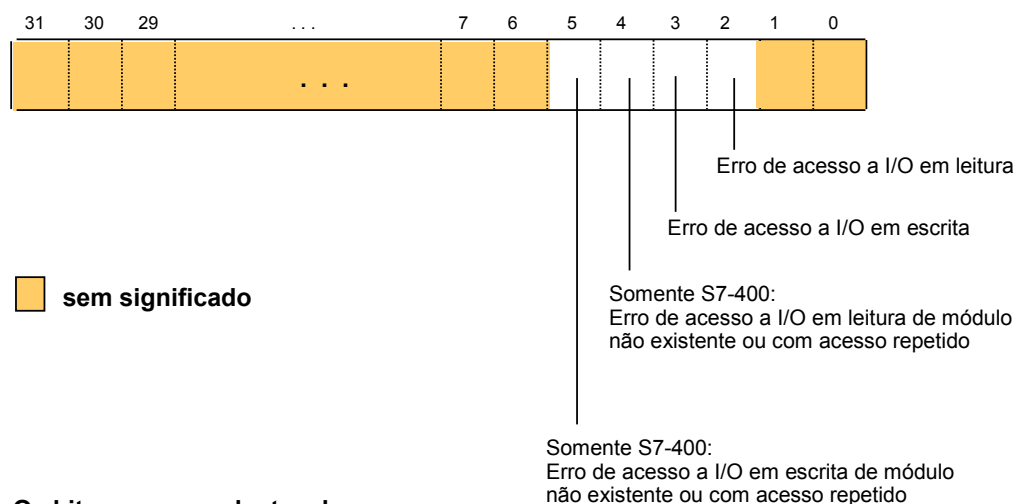
Conhecimento em Automação  
Training Center

### Programação de filtro de falha

Você controla a função do sistema para a manipulação de erro síncrono com os filtros de falha. Na programação do filtro de falhas existe um bit para cada possível falha de programação. Na especificação do filtro de falhas, você ajusta os bits de erro síncrono os quais você deseja mascarar, desmascarar ou checar.

O filtro de falhas envia pela função do sistema indicando com o nível lógico "1" os erros síncronos que ainda estão mascarados ou que ocorreram.

## Estrutura de acesso ao Filtro de Falhas



**Nota:** Os bits correspondentes do parâmetro de saída ACCFLT\_MASKED são setados como a seguir:

Valor = "1": Erro está mascarado.

Valor = "0": Erro não está mascarado.

Os bits não relevantes têm o valor "1".

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.11



Conhecimento em Automação  
Training Center

### Filtro de falhas de acesso

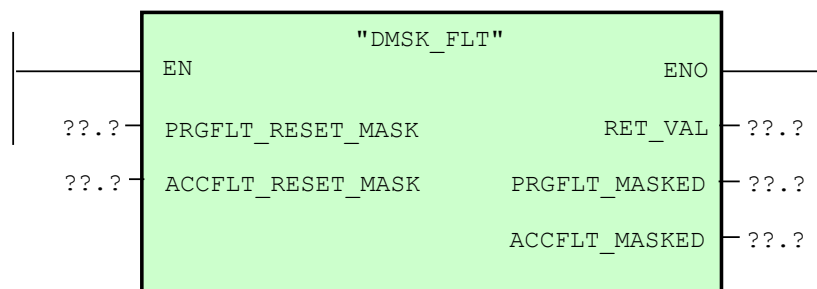
As CPUs S7-400 distinguem entre dois tipos de erro de acesso a periferia.

Acesso a módulo não existente e falha de acesso a módulo inserido como existente.

Se um módulo falha durante operação, um estouro de tempo (time-out) (QVZ) ocorre quando o módulo é acessado pelo programa. Ao mesmo tempo, este módulo é interpretado como "não existente", por tanto a cada acesso adicional um erro de acesso a periferia (I/O) (PZF) é sinalizado.

A CPU também sinaliza um erro de acesso a periferia (I/O) quando um módulo não existente é acessado, seja este diretamente através da área de I/O ou indiretamente através da imagem de processo.

## SFC 37 para desmascaramento de Falhas Síncronas



Parâmetro	Declaração	Tipo dado	Área de memória	Descrição
PRGFLT_RESET_MASK	INPUT	DWORD	I, Q, M, D, L, Const.	Programação filtro de falhas para resetar
ACCFLT_RESET_MASK	INPUT	BYTE	I, Q, M, D, L, Const.	Acesso ao filtro de falhas para resetar
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor do SFC, W#16#0001: o novo filtro contém bits que não são setados no filtro armazenado
PRGFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Mantém mascarados erros programação
ACCFLT_MASKED	OUTPUT	DWORD	I, Q, M, D, L	Mantém mascarados erros de acesso

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.12Conhecimento em Automação  
Training Center

### Desmascaramento de falhas síncronas

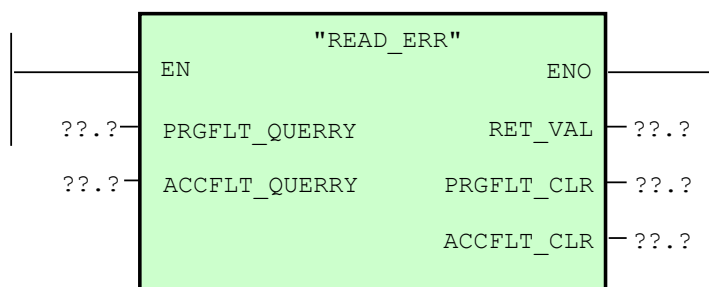
A função do sistema SFC37 (DMSK\_FLT) utiliza o filtro de falhas para habilitar a chamada dos OBs de erros síncronos de novo. Com o nível lógico "1" você identifica no filtro de falhas para quais erros síncronos os OBs estão de novo sendo chamados (as falhas síncronas são "desmascaradas"). As entradas correspondentes do desmascaramento especificado, que estão no registrador de erros, são apagadas.

No retorno do valor, a SFC37 sinaliza com W#16#0001 se para o desmascaramento especificado nos parâmetros de entrada, não existe máscara (armazenada) para pelo menos um bit.

A SFC37 entrega em seus parâmetros de saída todos os eventos mascarado atualmente com nível lógico "1".

Se uma falha síncrona desmascarada ocorre, o OB correspondente é chamado de novo e o evento é inserido no registrador de erro. Habilitação é válida para a classe de prioridade corrente.

## SFC 38 para leitura do Registrador de Erros



Parâmetro	Declaração	Tipo dado	Área de memória	Descrição
PRGFLT_QUERRY	INPUT	DWORD	I, Q, M, D, L, Const.	Programação filtro de falhas para checagem
ACCFLT_QUERRY	INPUT	BYTE	I, Q, M, D, L, Const.	Acesso filtro de falhas para checagem
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Retorna valor do SFC, W#16#0001: o filtro de verificação contém bits que não são setados (no filtro armazenado)
PRGFLT_CLR	OUTPUT	DWORD	I, Q, M, D, L	Programação filtro de falhas com mensagens de erro
ACCFLT_CLR	OUTPUT	DWORD	I, Q, M, D, L	Acesso filtro de falhas com mensagens de erro

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.13Conhecimento em Automação  
Training Center

### Leitura do registrador de erro

A função do sistema SFC38 (READ\_ERR) faz a leitura do registrador de erro. Com nível lógico "1" você identifica no filtro de falhas para quais erros síncronos você deseja que sejam lidas as entradas.

No retorno do valor, a SFC38 sinaliza com W#16#0001 se para a seleção especificada nos parâmetros de entrada, não existe máscara (armazenada) para pelo menos um bit.

A SFC38 retorna os eventos selecionados com nível lógico "1" nos parâmetros de saída, quando eles ocorrem e apaga estes eventos do registrador de erro com a varredura. Um bit setado significa que o erro síncrono associado mascarado ocorreu pelo menos uma vez.

As falhas síncronas que ocorreram na classe de prioridade corrente são sinalizadas.

## Exemplo: Testando Bloco de Dados

```

Network 1: Mascaramento, Teste, Desmascaramento
// Mascara "DB não existe"
Call SFC 36(
  PRGFLT_SET_MASK      := DW#16#4000000, // Identificador: DB não existe
  ACCFLT_SET_MASK      := DW#16#0,       // na máscara para erros de acesso
  RET_VAL              := #SFC36Error,
  PRGFLT_MASKED        := #Prog36Mask,
  ACCFLT_MASKED        := #Acc36Mask);

// Testa chamada
OPN DB[DB_NO];

// Checa programação de erro
Call SFC 38(
  PRGFLT_QUERRY        := DW#16#4000000, // Identificador: DB não existe
  ACCFLT_QUERRY        := DW#16#0,       // na máscara para erros de acesso
  RET_VAL              := #SFC38Error,
  PRGFLT_MASKED        := #Prog38Mask,
  ACCFLT_MASKED        := #Acc38Mask);

// Avalia resultado
L      #Prog38Mask
L      DW#16#4000000
==D
=      #DB_NOT_THERE // Seta variável auxiliar "DB não existe"

// Desmascara "DB não existe"
Call SFC 37(
  PRGFLT_RESET_MASK    := DW#16#4000000, // Identificador: DB não existe
  ACCFLT_RESET_MASK    := DW#16#0,       // na máscara para erros de acesso
  RET_VAL              := #SFC37Error,
  PRGFLT_MASKED        := #Prog37Mask,
  ACCFLT_MASKED        := #Acc37Mask);

```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.14Conhecimento em Automação  
Training Center

### Exemplo

Este exemplo mostra o procedimento para mascarar a possível falha síncrona na abertura de um DB.

1. No primeiro passo, a instrução "crítica" OPN DB... é mascarada com a ajuda da SFC 36 (MSK\_FLT).
2. Depois dela, a instrução OPN DB[DB\_NO] é executada. Se o DB não está na memória de trabalho da CPU, então OB121 não é chamado neste caso.
3. Com a ajuda da SFC38 (READ\_ERR), o registrador de erro é lido e checado de forma que a instrução para abertura do DB tenha falhado ou não.

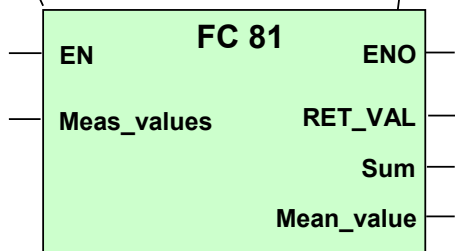
No caso de um erro, a variável local #DB\_NOT\_THERE é setada para "1" e só então uma avaliação pode ser feita posteriormente.

4. No fim, a falha síncrona mascarada é desmascarada de novo com a ajuda da SFC37 (DMSK\_FLT), através do reestabelecimento do estado original.

## Exercício 8.1: Manipulação de Erro no FC81

Decl.	Nome	Tipo
in	Meas_values	ANY
out	RET_VAL	INT
out	Sum	REAL
out	Mean_value	REAL

Causa	Código erro
Tudo O. K.	0
Tipo dado <>REAL	-1
DB não existe	-2
Erro compr. faixa	-4



Código de erro

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_08P.15



Conhecimento em Automação  
Training Center

### Vista geral

No exercício 4.3 você criou uma FC43 que determinava a soma e a média de valores de um ARRAY de números REAL. Até agora, somente um erro elementar manipulado (checando o tipo de dado) tem sido realizado dentro desta FC.

O erro manipulado está agora sendo expandido de modo que com a nova FC81 está "seguro contra colisão", isto é, com atribuição de parâmetro incorreto, nenhuma falha síncrona é gatilhada.

Mais adiante, a FC81 permite, no parâmetro de saída adicional #RET\_VAL, informação sobre o tipo de erro.

### Objetivo

Antes de tudo copia a FC43 dentro da FC81 e integra a seguinte manipulação de erro :

- Se um tipo de dado diferente de REAL é passado, então a FC81 é excitada com um código de erro -1.
- Se um número inválido de DB é passado (p.ex. número fora da faixa permitida DB não existente), então FC81 é excitada com um código de erro -2.
- Se dentro de um loop existe um acesso a um endereço não existente (erro de faixa ou comprimento de faixa), então FC81 é excitada com um código de erro -4.
- Em todos os casos de erro, FC81 ajusta o bit BR em zero e retorna um número REAL inválido no parâmetro de saída #Sum e #Mean\_value.

### O que fazer

1. Complementar FC81 com o parâmetro de saída #RET\_VAL (código erro).
2. Na FC81 implementar o correspondente erro manipulado.
3. Programar a chamada da FC81 no OB1.
4. Transferir os blocos participantes para a CPU e testar o resultado.

### Questão?

Como você pode fazer a "colisão" na FC81?

## Geração de Programas com o Editor de Textos



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.1



Conhecimento em Automação  
Training Center

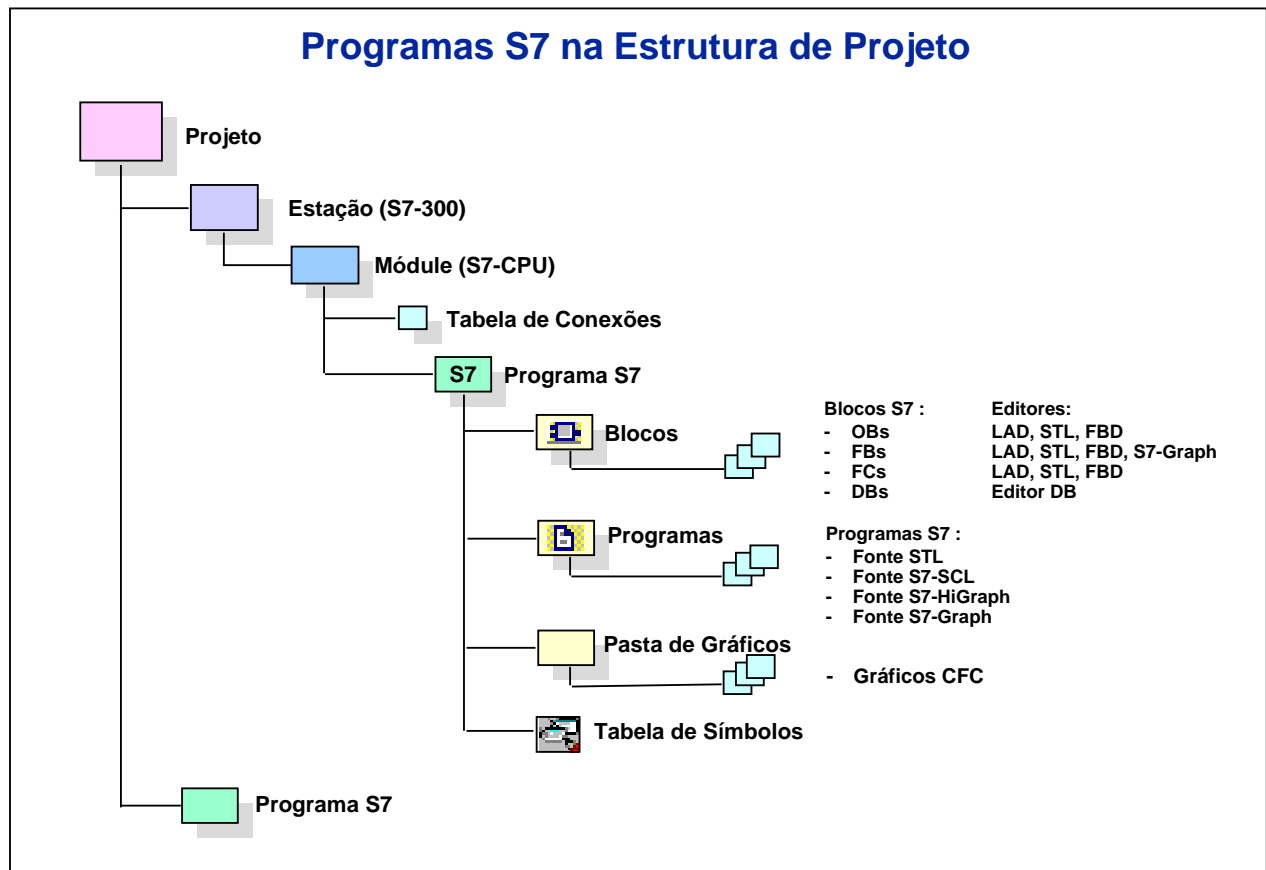
### Conteúdo

### Pág.

Programas S7 na Estrutura de Projeto .....	2
Conceito de Entrada e Compilação .....	3
Iniciando o Editor de Textos .....	4
Geração de Programas com o Editor de Textos .....	5
Inserindo Blocos Templates, Blocos e Programas .....	6
Regras de Entrada Geral e Estrutura .....	7
Sintaxe para Blocos Lógicos .....	8
Sintaxe para Blocos de Dados .....	9
Regras para Declaração de Variáveis .....	10
Alocação de Atributos de Blocos .....	11
Exercício 9.1: Criando um Arquivo Fonte .....	12
Exercício 9.2: Contagem de peças acabadas .....	13



## Programas S7 na Estrutura de Projeto



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.2



Conhecimento em Automação  
Training Center

### Vista Geral

De forma a possibilitar a criação de um novo programa S7, antes de mais nada um projeto deve primeiro ser gerado no SIMATIC Manager. Subseqüentemente, existem duas possibilidades para criação de uma pasta de programa S7:

- **Módulos independentes:** Neste caso você deve inserir a pasta de programa para programas S7 diretamente ligada ao ícone do Projeto. Os programas criados nela podem depois serem atribuídos a um módulo programável..
- **Módulo dependente:** Neste caso o projeto deve conter pelo menos uma estação SIMATIC 300/400 com um módulo programável (CPU). Uma pasta de programa S7 é então automaticamente inserida ligada ao ícone do módulo programável.

Se você deseja usar símbolos globais em seu programa do usuário, você deve fazer a atribuição correspondente dos identificadores e endereços absolutos na tabela de símbolos antes de utilizá-los.

### Blocos, Fontes e Gráficos

Você pode guardar o programa S7 como programa de usuário (bloco), arquivos fonte ou gráficos. Programas e gráficos são somente utilizados contudo, como base para geração de programas em S7. Somente blocos podem ser transferidos para uma CPU S7.

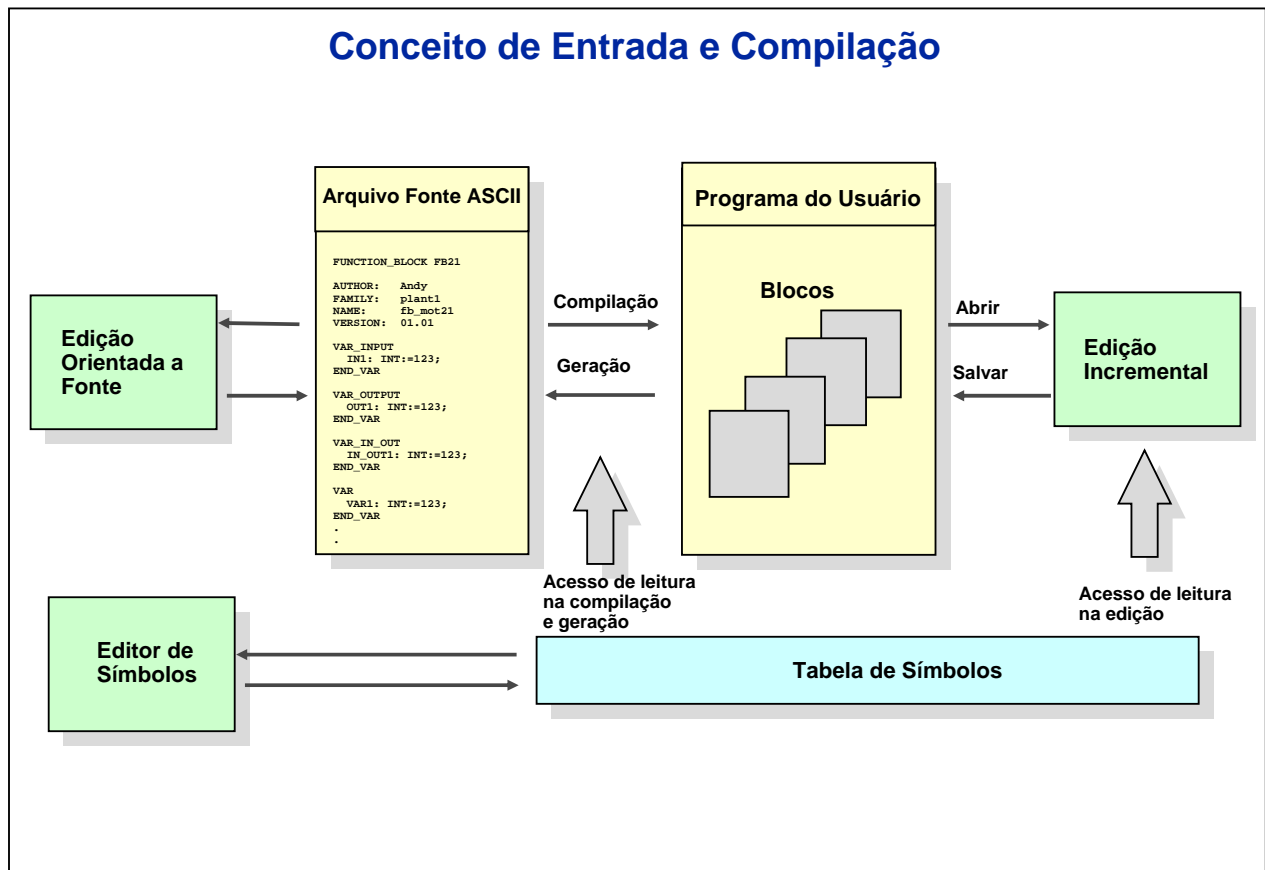
Se você gerar um bloco, um arquivo fonte ou um gráfico, dependendo da linguagem de programação selecionada ou da linguagem do editor.

**Programa do Usuário** Somente os blocos do programa do usuário pode ser transferido para uma CPU S7. Dependendo do escopo, isto inclui blocos de organização (OBs), funções (FCs), blocos de funções (FBs) e blocos de dados (DBs).

O tipo de dado definido pelo usuário (UDTs) criado simplifica simplesmente a programação, ele não pode portanto ser transferido para uma CPU S7.

O mesmo é válido para a tabela de variáveis (VATs), na qual endereços para funções *Monitor/Modify Variables* são salvas.

## Conceito de Entrada e Compilação



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.3



Conhecimento em Automação  
Training Center

### Possibilidades de Entrada

Dependendo da linguagem de programação que você tem escolhido para geração de programa, você pode entrar seu programa incrementalmente e/ou orientado a fonte.

- Entrada incremental (STL, LAD, FBD, S7-Graph, S7-HiGraph, CFC)

Cada linha ou cada elemento é imediatamente examinado pelos erros de sintaxe após a entrada. Caso ocorram erros de entrada, estes serão indicados (marcados em vermelho) e devem ser corrigidos antes de salvar.

A sintaxe correta das entradas são automaticamente compiladas e mostradas em preto. Em entradas incrementais, os símbolos utilizados já devem ser definidos na tabela de símbolos, de outra forma a entrada será marcada em vermelho e uma mensagem de erro correspondente será mostrada na barra de estado (status bar).

- Entrada orientada a fonte (STL, S7-SCL)

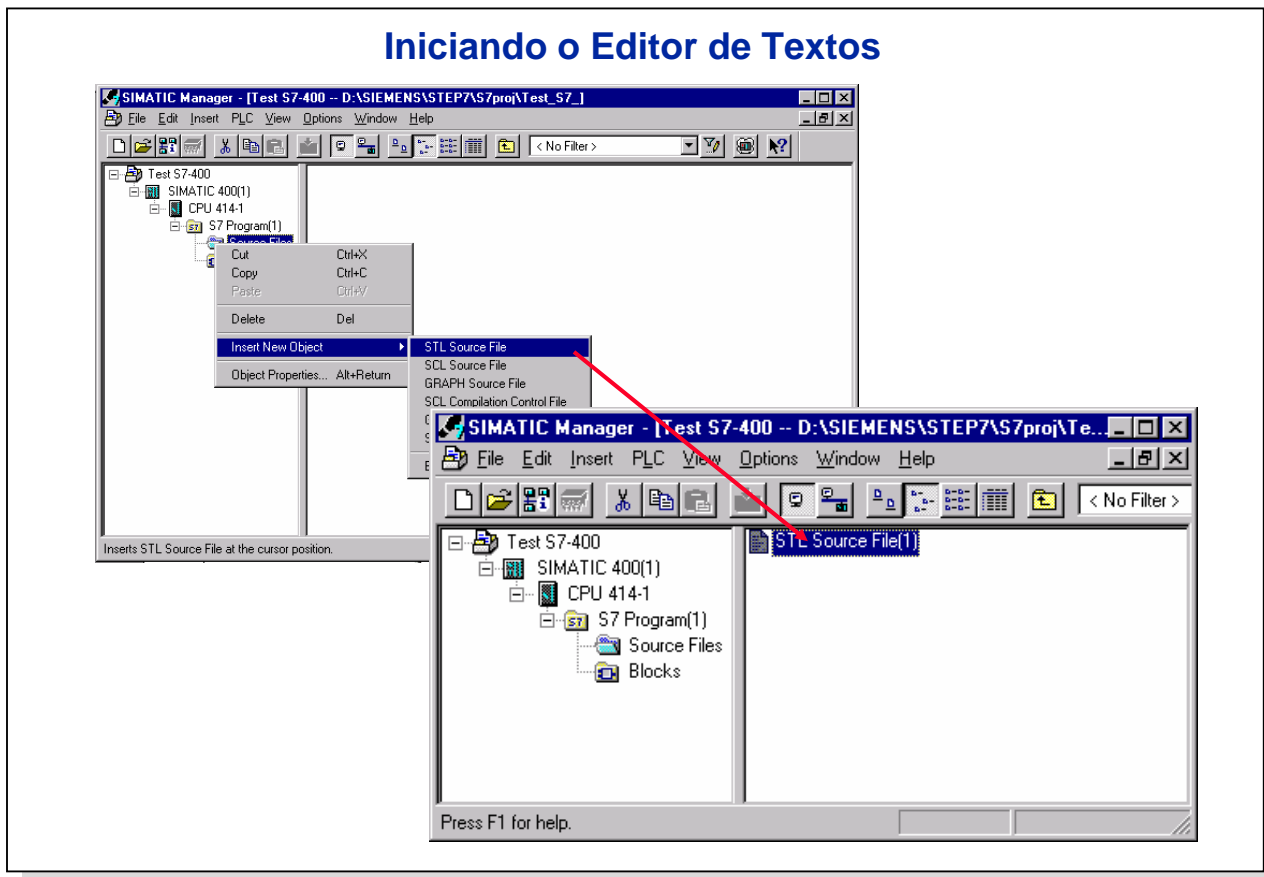
Na entrada orientada a fonte, o programa ou um bloco é editado em um arquivo texto e o arquivo texto é então compilado, onde os erros são primeiro indicados na compilação pelo Compilador associado.

Na entrada orientada a fonte, os símbolos somente devem ser definidos na tabela de símbolos no momento da compilação. Programas tem a vantagem que eles podem ser exportados - então processados com as ferramentas da escolha - e podem então ser reimportados.

### Vantagens da entrada orientada a fonte

- Diversos blocos podem ser armazenados em um arquivo fonte (os blocos devem, deste modo, ser armazenados de forma a que os blocos chamados estejam sempre localizados antes dos blocos chamados).
- Um arquivo fonte pode ser salvo com erros de sintaxe.
- Você pode gerar seu arquivo fonte com outros editores, importa-los para o SIMATIC Manager e então compila-los transformando-os em blocos.
- Um bloco protegido somente pode ser entrado no modo ASCII.
- Mudanças na chamada de blocos aninhados (p.ex. adição de parâmetros de blocos) podem ser melhor manipulados com o Editor ASCII (p.ex. com Find e Replace), do que com o Editor Incremental.

## Iniciando o Editor de Textos



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.4



Conhecimento em Automação  
Training Center

### Iniciando do SIMATIC Manager

Você parte o Editor de Textos do SIMATIC Manager. Pré-requisito é que você tenha criado um projeto com um programa S7. Você pode gerar o programa dependente ou independente do hardware.

Com o Editor de Textos você processa exclusivamente programas, os quais você subsequenteiramente compila para blocos, estes são armazenados na pasta de Blocos.

### Criando uma Fonte

Quando você deseja gerar um novo arquivo fonte, em primeiro lugar você deve antes de mais nada criar um arquivo vazio no SIMATIC Manager através do qual você abre com o Editor de Textos. Quando você tiver aberto o Editor você pode criar os programas neste.

- No SIMATIC Manager selecione a pasta de programas e insira um arquivo com a opção menu *Insert New Object -> STL Source File*. O novo arquivo fonte aparece no lado direito da janela projeto com um nome presetado.
- No Editor de Textos, você pode simplesmente criar um novo arquivo fonte usando a opção menu *File -> New*. No diálogo seguinte você entra com o nome do novo arquivo fonte.

### Abrindo um Arquivo Fonte

Você abre um arquivo fonte no SIMATIC Manager através de um duplo clique no seu símbolo. Alternativamente, você pode chegar a este utilizando a opção menu *Edit -> Open Object* ou o ícone correspondente na barra de ferramentas.

### Gerando um Arquivo Fonte

Também é possível reverter a compilação de blocos já existentes em um arquivo fonte, de forma a permitir posterior processamento. No Editor de Textos selecione a opção de menu *File -> Generate Source* para tanto. No diálogo seguinte você pode selecionar todos os blocos os quais você deseja gerar um arquivo fonte.

## Geração de Programas com o Editor de Textos

```

LAD/STL/FBD - [EXERCISE_5.1 -- Pro2_e_solution\CHAP_05]
File Edit Insert PLC Debug View Options Window Help

// Address assignment in FC51 is adopted to S7-300 16 bit

FUNCTION FC 51 : VOID
TITLE = Exercise 5.1: Read System Clock
//Version for 16Bit-SM
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
    Date_Time : DATE_AND_TIME ;           //Current Time and Date
    RET_VAL_SFC1 : INT ;                   //Return value of SFC 1
END_VAR
BEGIN
NETWORK
TITLE =Call SFC 1 (READ_CLK)

    CALL SFC1 (
        RET_VAL           := #RET_VAL_SFC1,
        CDT                := #Date_Time);

    NOP    0;
NETWORK
TITLE =Display hours and minutes

    LAR1 P##Date_Time;           // Get address of #Date_Time
    L    LB [AR1, P#3.0];        // Read hours
    T    QB    12;               // and transfer to diq. display
  
```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.5



Conhecimento em Automação  
Training Center

### Editor de Textos

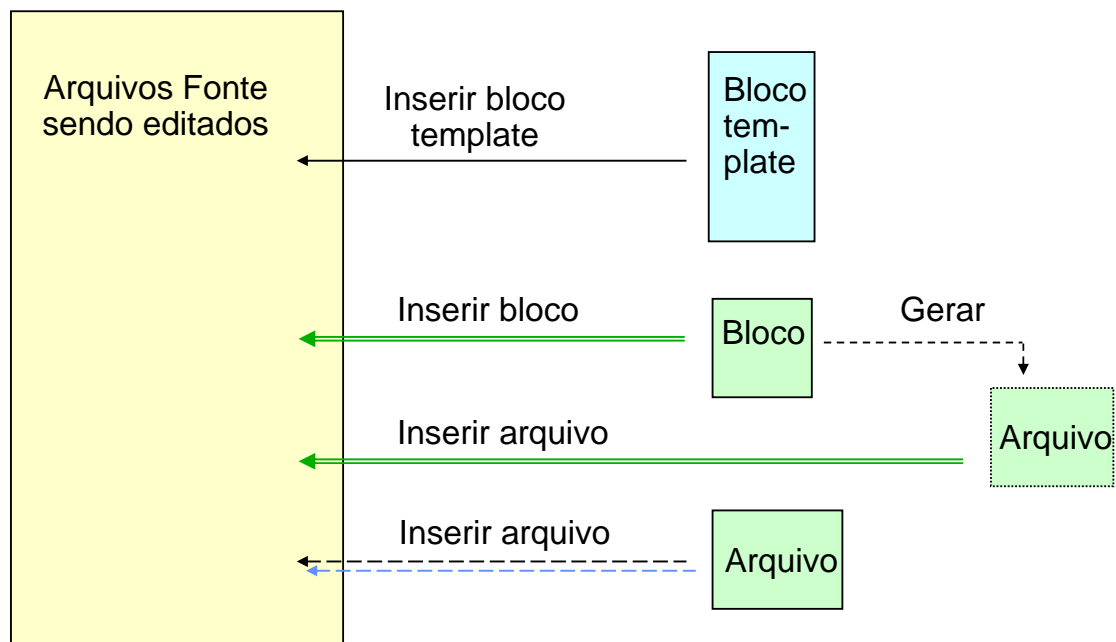
Em vez de programar em STL, você pode gerar seu programa com o Editor de Texto integrado e deste modo criar um arquivo fonte. Você entra com seus blocos um após o outro (possibilidade de diversos blocos em um arquivo fonte). Uma verificação da sintaxe não ocorre.

### Ajustes

Antes de você começar a programar no Editor de Textos, você deve começar a se familiarizar com as possibilidades de customização, de forma a permitir trabalhar confortavelmente e de acordo com suas preferências pessoais.

Você abre um diálogo de registro utilizando a opção de menu *Options Settings*. No registro "Editor" você pode ajustar o padrão de script (tipo e tamanho) para o arquivo fonte. A cor na qual linhas de instruções são marcadas é mudada no registro "LAD/FBD".

## Inserindo Templates de Blocos, Blocos e Programas



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.6



Conhecimento em Automação  
Training Center

#### Inserindo Blocos Templates

Blocos templates para OBs, FBs, FCs, DBs, DBs Instance, DBs de UDTs e UDTs são integradas no Editor para programação simplificada. Um bloco template contém todas as palavras chave requeridas na sequência necessária. Você simplesmente deleta os templates de declarações opcionais as quais você não deseja fazer. Os templates de blocos facilitam a entrada e aderência para sintaxe e estrutura ao mesmo tempo.

De forma a inserir um template de bloco em seu arquivo fonte, selecione a opção de menu *Insert -> Block Template -> OB/FB/FC/DB/IDB/ DB from UDT/UDT*.

#### Inserindo Blocos

Você pode inserir em seu arquivo fonte os códigos fonte correspondentes dos blocos que já havia sido gerado. Para isto, selecione a opção de menu *Insert -> Object -> Block*. No diálogo seguinte selecione os blocos cujos códigos você deseja inserir como texto.

Um arquivo fonte é gerado de blocos selecionados. Seus conteúdos são inseridos depois da posição do cursor no arquivo fonte onde está sendo editado.

#### Inserindo Arquivos Fontes

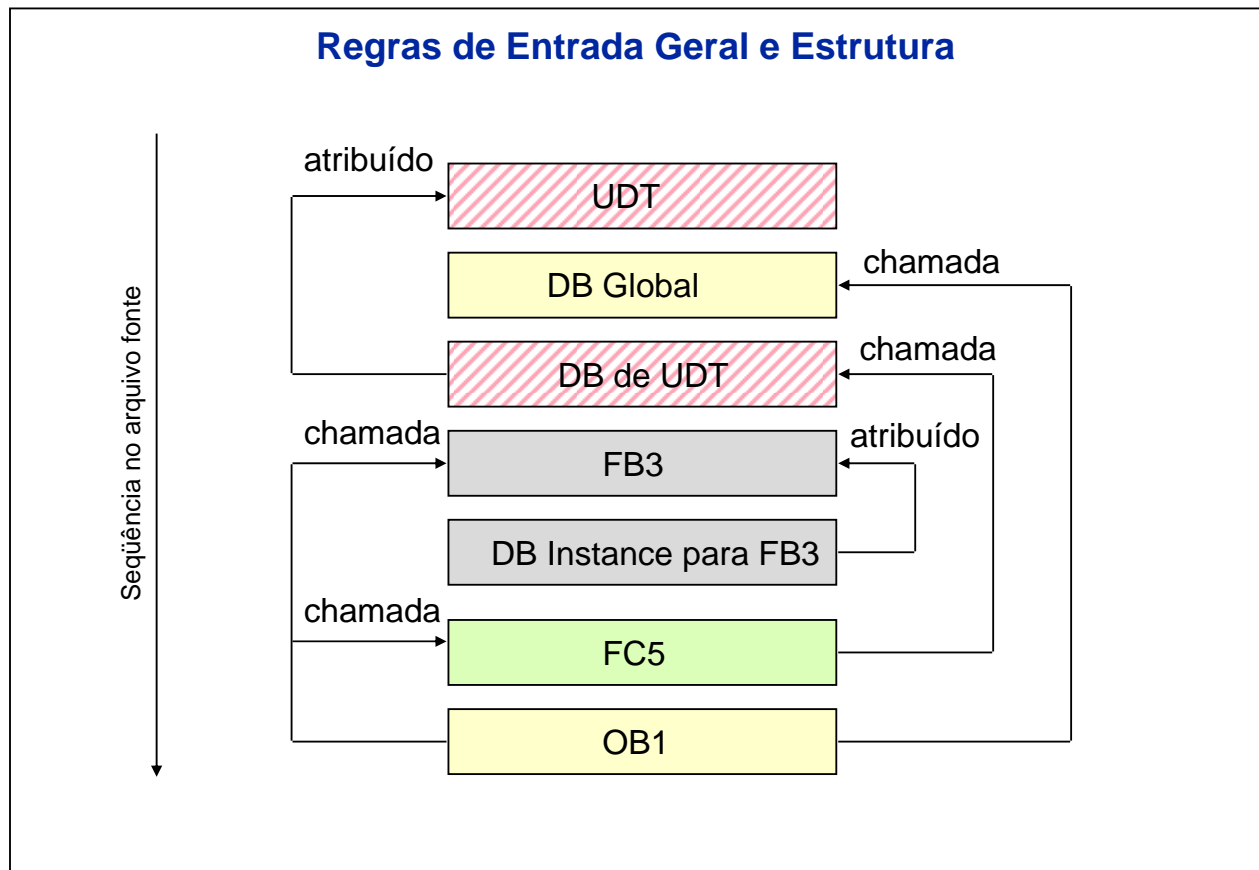
Você pode inserir o conteúdo de qualquer outros programas em seu arquivo fonte. Para isto, selecione a opção de menu *Insert -> Object -> File* e no diálogo seguinte selecione o arquivo a ser inserido.

Desta forma, o conteúdo de qualquer arquivo texto pode ser inserido em seu arquivo fonte.

#### Nota

Qualquer conteúdo de texto pode também ser inserido em seu arquivo fonte usando o clipboard Windows.

## Regras de Entrada Geral e Estrutura



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.7Conhecimento em Automação  
Training Center

### Regras de Entrada



As seguintes regras gerais são válidas para a geração dos programas do usuário como arquivo fonte:

- A sintaxe das instruções STL é o mesmo que no Editor STL incremental. Existe uma exceção com a chamada dos blocos e a declaração dos Arrays e Structs.
- Em geral, o Editor de Textos não levar em conta letras maiúsculas e minúsculas. A exceção são os rótulos de saltos.
- Identifica o fim de cada instrução STL e cada declaração de variável com ponto e vírgula (;). Você pode entrar mais do que uma instrução por linha.
- Começar cada comentário com duas barras (//) e terminar cada entrada de comentário com a tecla ENTER.

### Sequência de Bloco

Com relação a sequência dos blocos, você deve prestar atenção ao seguinte na geração do arquivo fonte:

Chamadas de blocos são locadas antes dos blocos chamados. Isto significa:

- OB1, o qual é utilizado mais frequentemente e o qual chama os outros blocos, fica por último. Os blocos, os quais são chamados pelos blocos que são chamados no OB1, devem estar antes deste, etc.
- Tipos de dados definidos pelo usuário (UDTs) ficam antes dos blocos nos quais eles são utilizados.
- Blocos de dados que atribuem tipos de dados definidos pelo usuário (UDT) ficam após o UDT.
- Blocos de dados Globais ficam antes de todos os blocos dos quais eles são chamados.
- Blocos de dados que atribuem blocos de funções (DB instance) ficam depois do bloco de funções.

## Sintaxe para Blocos Lógicos

Configuração	Palavra chave com Exemplo
Início do bloco com especificação do bloco (absoluto ou simbólico)	ORGANIZATION_BLOCK OB1 FUNCTION_BLOCK FB1 FUNCTION FC 1 : int
Título de bloco (opcional)	TITLE = <i>Block title</i>
Comentário de bloco (opcional)	// <i>Block comment</i>
Atributos de Sistema para bloco (opcional)	{Attr1 := 'block_val1'; // Block attribute1 Attr2 := 'block_val2'; // Block attribute2 Attr3 := 'block_val3' // Block attribute3}
Propriedades de bloco (opcional)	KNOW_HOW_PROTECT AUTHOR: PT41 FAMILY: Motors NAME: Motorone VERSION: 0815
Declaração das partes variáveis (tipo de declaração, dependendo do tipo de bloco)	VAR_IN VAR_OUT VAR_IN_OUT VAR VAR_TEMP ..
Término de cada tipo de declaração com	END_VAR
Parte das instruções consistindo de Networks com Título de Network Comentário de Network	BEGIN NETWORK TITLE= <i>first network</i> //
Fim de bloco	END_ORGANIZATION_BLOCK END_FUNCTION_BLOCK END_FUNCTION

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.8Conhecimento em Automação  
Training Center

### Regras

Com a entrada de um bloco lógico, você deve prestar atenção nas seguintes regras:

- No início do bloco, existe um espaço entre a palavra chave para o tipo de bloco e a especificação do bloco. Na especificação do nome simbólico do bloco, você pode identifica-lo entre aspas, isto para garantir a diferenciação entre nomes de variáveis locais e nomes da tabela de símbolos.
- Com funções (FCs), o tipo de função é igualmente fornecido. Isto pode ser tipos de dados elementares ou complexos e determinar o tipo de dado do valor de retorno (#RET\_VAL). Se nenhum valor está sendo retornado, VOID está sendo inserido.
- A especificação do número de network não é permitido.

### Chamada de Blocos com "CALL"

A sintaxe para a chamada dos FBs e FCs com o comando CALL desvia-se levemente deste no Editor STL incremental. Em um arquivo fonte você entra com os parâmetros entre parêntesis. Os parâmetros individuais são então separados um do outro por vírgulas.

Exemplo: CALL FC1 (param1 := I 0.0, param2 := I 0.1);

### Comentários na Parte das Instruções

De forma a garantir uma representação um pra um dos comentários na última edição no Editor incremental, você deve prestar atenção no seguinte:

- Chamada de Bloco: Em programas, você deve guardar a seqüência dos parâmetros formais como ele são na declaração de variáveis do bloco quando você atribui parâmetros atuais para os parâmetros formais. Embora a seqüência dos parâmetros seja escolhida, comentários para os parâmetros podem deste modo ser alterados durante a compilação da fonte em blocos.
- Com instruções para acessar blocos de dados que diretamente seguem a instrução "OPN", isto é possível que uma perda dos comentários da instrução possa ocorrer durante a compilação para bloco. De forma a evitar isto, programe de forma compacta (p.ex. L DB5.DBW20; //Comentário) ou escreva em uma instrução "NOP" (p.ex. OPN DB5; //Comentário 1 NOP 0; L DBW20; //Comentário 2).



## Sintaxe para Blocos de Dados

Configuração	Palavra chave com Exemplo
Início do bloco com especificação do bloco (absoluto ou simbólico)	DATA_BLOCK DB 26
Título do bloco (opcional)	TITLE = <i>Block title</i>
Comentário do bloco (opcional)	// <i>Block comment</i>
Atributos do Sistema para blocos (opcional)	{Attr1 := 'block_val1'; // Block attribute1 Attr2 := 'block_val2'; // Block attribute2}
Propriedades dos blocos (opcional)	KNOW_HOW_PROTECT AUTHOR: <i>Müller</i> FAMILY: <i>Motors</i> NAME: <i>Motorone</i> VERSION: <i>0815</i>
Parte da declaração – dependendo do DB	
Bloco de dados Global: Declaração de Variáveis (opcional com valores iniciais)	STRUCT .. .. END_STRUCT
DB de UDT: Especificação da UDT (absoluta ou simbólica)	UDT 16
DB Instance : Especificação do FB (absoluto ou simbólico)	FB 20
Parte de atribuição com valores correntes	BEGIN ..
Fim de Bloco	END_DATA_BLOCK

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.9Conhecimento em Automação  
Training Center

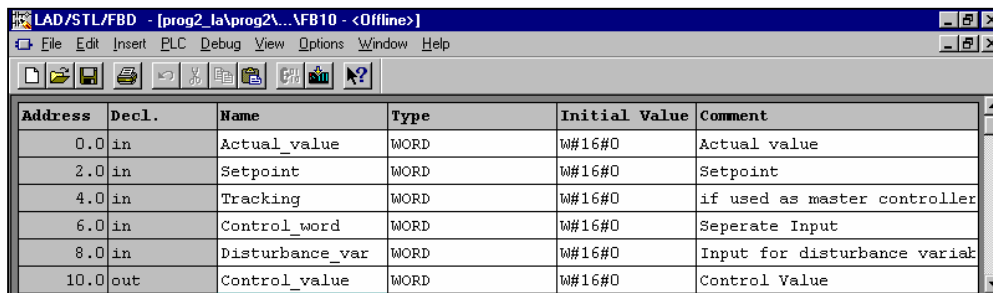
### Regras

Na entrada de dados dos blocos de dados você deve prestar atenção às seguintes regras:

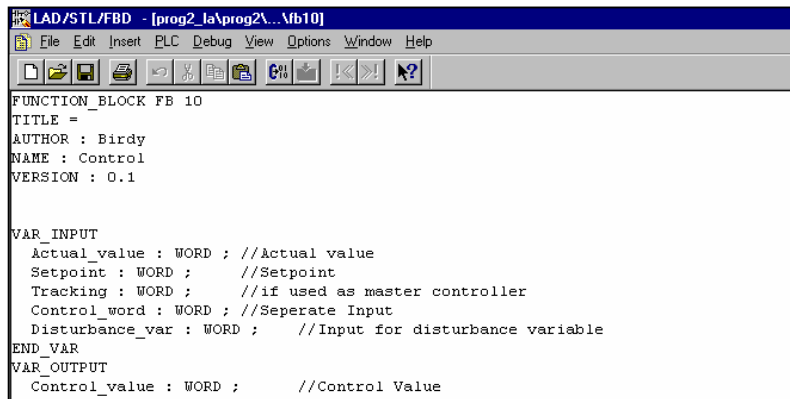
- Você não pode gerar um DB 0.
- Você pode opcionalmente especificar valores correntes para todas ou algumas variáveis. Para variáveis, as quais você não atribuiu valores correntes, o valor inicial é atribuído, se disponível, por outro lado valores padrões dos tipos de dados são atribuídos.
- Comentários de instruções na seção de atribuição para valores correntes (entre BEGIN e END\_DATA\_BLOCK) não são mostradas no Editor incremental após compilação para blocos. Por esta razão somente escreva comentários para blocos de dados na parte da declaração.



## Regras para Declaração de Variáveis



Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Actual_value	WORD	W#16#0	Actual value
2.0	in	Setpoint	WORD	W#16#0	Setpoint
4.0	in	Tracking	WORD	W#16#0	if used as master controller
6.0	in	Control_word	WORD	W#16#0	Seperate Input
8.0	in	Disturbance_var	WORD	W#16#0	Input for disturbance variak
10.0	out	Control_value	WORD	W#16#0	Control Value



```

FUNCTION_BLOCK FB 10
TITLE =
AUTHOR : Birdy
NAME : Control
VERSION : 0.1

VAR_INPUT
    Actual_value : WORD ; //Actual value
    Setpoint : WORD ; //Setpoint
    Tracking : WORD ; //if used as master controller
    Control_word : WORD ; //Seperate Input
    Disturbance_var : WORD ; //Input for disturbance variable
END_VAR
VAR_OUTPUT
    Control_value : WORD ; //Control Value
  
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

 Date: 04.10.2007  
 File: PRO2\_09P.10

 Conhecimento em Automação  
 Training Center

### Tipos de Variáveis

Com blocos lógicos, o tipo de declaração das variáveis é identificado por uma palavra chave, isto é encontrado em sua própria linha. Dependendo do tipo de bloco, somente tipos de declarações particulares são permitidas.

Tipo de Declaração	Palavra chave	OB	FB	FC
Parâmetro de entrada	VAR_INPUT	-	sim	sim
Parâmetro de saída	VAR_OUTPUT	-	sim	sim
Parâmetro entrada/saída	VAR_IN_OUT	-	sim	sim
Variável Estática	VAR	-	sim	-
Variável Temporária	VAR_TEMP	sim	sim	sim
Cada término com	END_VAR			

### Regras de Entrada

Na entrada de dados na declaração de variáveis, você deve prestar atenção no seguinte:

- As variáveis devem ser declaradas na seqüência dos tipos de declarações. Todas as variáveis de um tipo estão deste modo juntas.
- As palavras chaves são encontradas em cada caso em sua própria linha ou são separadas por um espaço em branco.
- O nome das variáveis é encontrado no início da linha e deve começar com uma letra. Ele não pode corresponder a qualquer das palavras chaves.
- Atributos de sistema opcional podem ser atribuídos para parâmetros individuais após o nome da variável. Os atributos do sistema são encerrados entre parênteses.

Exemplo: Var\_1 { ident1 := 'string1' ; ident2 := 'string2' } : INT;

Var\_2 { message := 'TRUE' ; OPERABLE := 'TRUE' } : INT;

- O tipo de dado é fornecido, separado por uma vírgula, depois do nome ou após o atributo do sistema. Tipos de dados elementares, complexos e definidos pelo usuário são permitidos.
- Cada declaração de variável é terminada por um ponto e vírgula.
- Comentários são separados de uma parte declaração por duas barras.

## Alocação de Atributos de Blocos

Atributo	Blocos Lógicos (OB, FB, FC)	Blocos de Dados	UDT
KNOW_HOW_PROTECT	sim	sim	não
AUTHOR	sim	sim	não
FAMILY	sim	sim	não
NAME	sim	sim	não
VERSION	sim	sim	não
UNLINKED	não	sim	não
READ_ONLY	não	sim	não

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.11



Conhecimento em Automação  
Training Center

**Atributos do Sistema** Você pode atribuir atributos do sistema para blocos, por exemplo, para diagnósticos do processo ou configuração de controle do sistema. Eles controlam a configuração de mensagem e configuração de conexão, funções de interface do operador e a configuração de controle do sistema.

**Propriedades dos Blocos** Você pode especificar o nome do bloco, família, versão e autor com a ajuda das palavras chaves. Para isto é válido o seguinte:

- Propriedades dos blocos são especificadas antes da parte de declaração das variáveis.
- Não existe nenhum ponto e vírgula no fim da linha.

**Proteção de Bloco** Você pode setar proteção do bloco para blocos lógicos e blocos de dados, pela especificação da palavra chave KNOW\_HOW\_PROTECT :

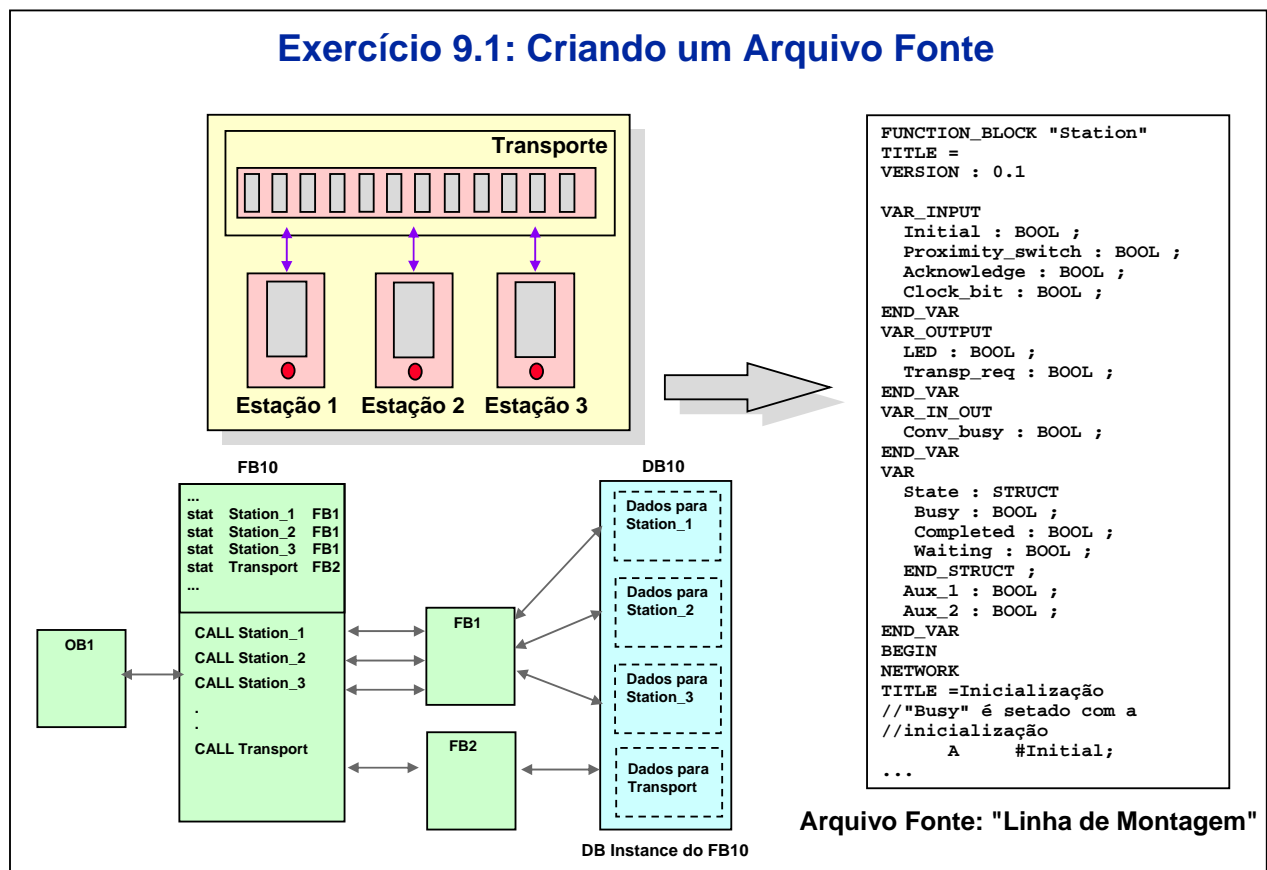
- Quando você procura o bloco compilado no Editor incremental STL, a parte de instruções de bloco não pode ser visto por dentro.
- Somente os parâmetros in, out e in/out são mostrados na declaração das variáveis dos blocos. As variáveis internas VAR e VAR\_TEMP permanecem ocultas.
- O bloco compilado pode ser compilado em um arquivo fonte, mas somente como um bloco sem a parte das instruções.

A palavra chave KNOW\_HOW\_PROTECT deve ser entrado antes de todos os atributos do bloco.

**Proteção de Escrita READ\_ONLY** Você pode setar uma proteção de escrita para blocos de dados em programas, desde que os valores dos dados armazenados dentro deles não podem ser sobrescritos durante a execução do programa. Para isto, entrar com a palavra chave READ\_ONLY . Este deve ser encontrado em sua própria linha diretamente antes das declarações.

**UNLINKED** O atributo UNLINKED somente pode ocorrer com blocos de dados. Ele diz que o DB não é carregado da memória de carga para a memória de trabalho da CPU.

## Exercício 9.1: Criando um Arquivo Fonte



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.12Conhecimento em Automação  
Training Center

### Vista Geral

Primeiro de tudo, um arquivo fonte está sendo criado de um programa final do exercício 6.2 "Linha de Montagem". Subseqüentemente, a funcionalidade de contagem adicional está sendo então introduzido dentro de blocos para a parte de transporte.

**Objetivo do exercício** Na pasta *programs* do projeto PRO2 (pasta de programa *Conv*), gerar um arquivo fonte, que inclui o programa do usuário total do exercício 6.2 e que pode ser compilado sem mensagens de erro.

### Procedimento

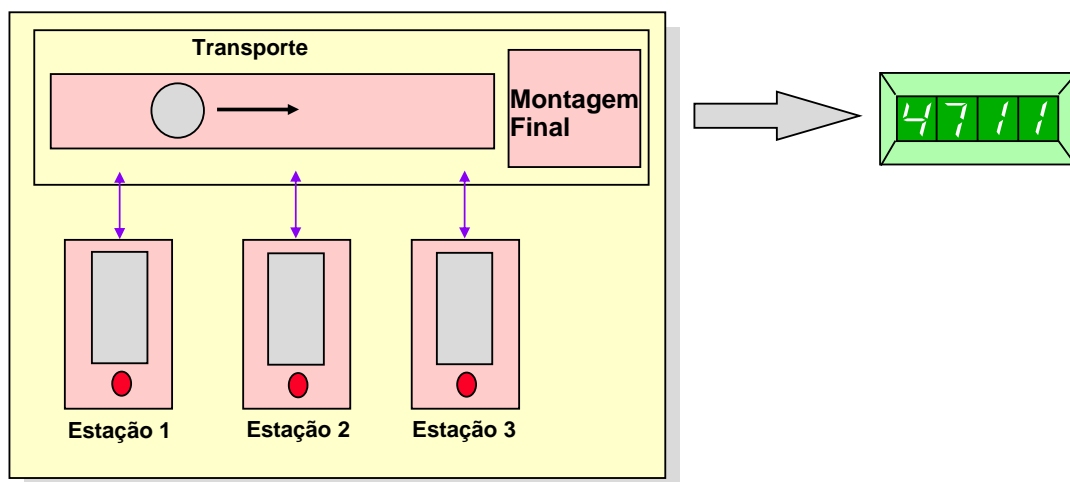
1. Antes de tudo, abra na pasta *Blocks* a pasta de programa *Conv* um bloco escolhido com a ajuda do Editor STL/LAD/FBD.
2. Então selecione a opção do menu *File -> Generate Source*. O diálogo "New" para entrada de dados o nome desejado do arquivo fonte aparece.
3. Entre com o nome do arquivo fonte (p.ex. Linha de montagem) e confirme o diálogo com "OK". O diálogo seguinte "Select STEP7 Blocks" aparece.
4. Selecione o bloco desejado e reconheça com "OK".

Note: Você pode selecionar a caixa de verificação "Program structur (XREF) sorted". Desta forma, os blocos são automaticamente arranjados na seqüência correta no arquivo fonte.

Geração do arquivo fonte é iniciado.

5. Com o Editor de Textos, abra o arquivo fonte criado.
6. Com a ajuda da opção do menu *File -> Check Consistency*, teste se o arquivo fonte pode ser compilado livre de erros.

## Exercício 9.2: Contagem de peças acabadas



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_09P.13



Conhecimento em Automação  
Training Center

**Objetivo do exercício** No bloco de funções "Transport", integra um contador que conta as peças completadas que chegam para montagem final. As propriedades do contador devem incluir as seguintes funcionalidades:

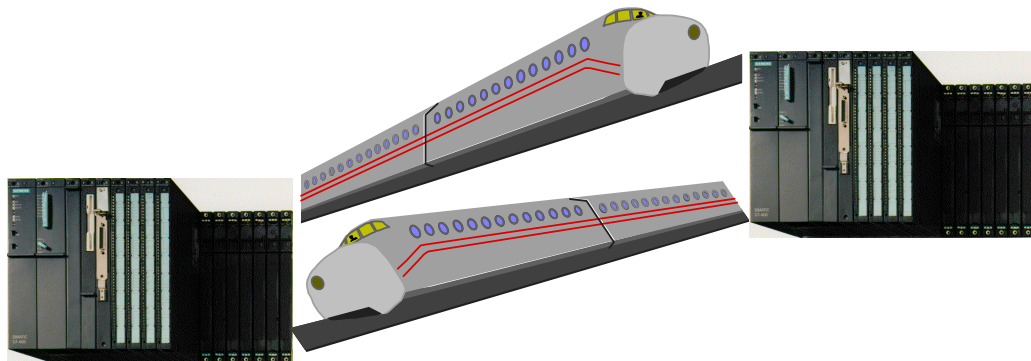
- O contador está sendo implementado com a ajuda do contador crescente (SFB 0 "CTU") conforme IEC 1131-3.
- Com cada transição negativa da barreira de luz, o contador incrementa, no estado #Transport\_right, sua contagem.
- O contador é resetado com o sinal de entrada #Initial.
- A contagem corrente é passada para o bloco chamado através de um parâmetro adicional de saída #Count\_Value (tipo de dado: INT).
- O valor de contagem é mostrado no display digital do simulador.
- Programe todos os passos do programa exclusivamente no arquivo fonte.
- Insira uma proteção de bloco em todos os FBs e DBs utilizando a palavra chave KNOW\_HOW\_PROTECT.

### O que fazer

1. Copie o SFB 0 da biblioteca Standard Library V3.x ou o FB6 dentro da sua pasta de bloco.
2. Abra o arquivo fonte *Assembly line (Linha de montagem)*.
3. No FB2 "Transport", declare a variável estática #Counter do tipo de dado SFB 0 ou FB6 bem como parâmetro de saída #Count\_Value do tipo de dado INT.
4. Insira as instruções necessárias da função de contagem no FB2 "Transport".
5. No FB10 insira as instruções para mostrar o valor de contagem (codificado em BCD) no display digital.
6. Compile o arquivo fonte modificado e transfira os novos blocos para a CPU. Teste o programa.
7. Insira uma proteção de bloco em todos os FBs e DBs participantes.



## Comunicação Básica e Expandida S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.1



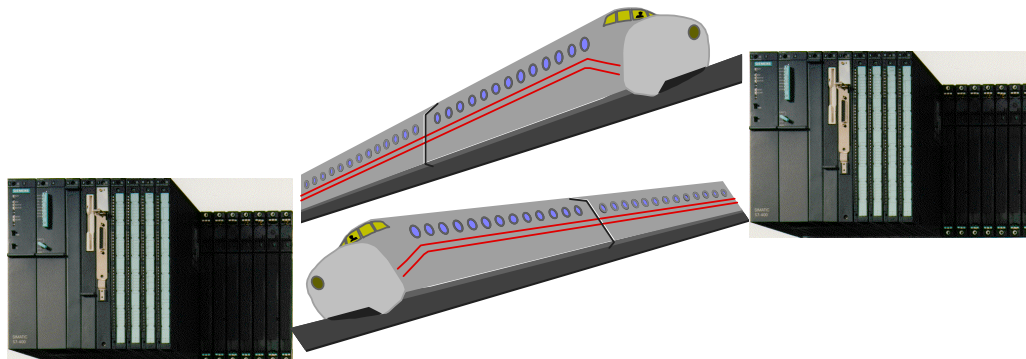
Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Sub-redes em SIMATIC .....	3
Serviços de Comunicação para SIMATIC .....	4
Serviços de Comunicação S7 para S7-300/400 .....	5
Conexões entre Participantes da Comunicação .....	6
Atribuição de Recursos de Conexão para Comunicação S7 .....	7
Dados Característicos das CPUs S7 - Comunicação .....	8
SFCs de Comunicação: Vista Geral .....	9
SFCs de Comunicação: Vista Geral dos Blocos .....	10
SFCs de Comunicação: Bloco X_GET (SFC 67) .....	11
SFCs de Comunicação: Bloco X_PUT (SFC 68) .....	12
SFCs de Comunicação: Bloco X_SEND (SFC 65) .....	13
SFCs de Comunicação: Bloco X_RCV (SFC 66) .....	14
SFBs de Comunicação: Vista Geral .....	15
SFBs de Comunicação: Vista Geral dos Blocos .....	16
Serviços de Comunicação de "mão única" utilizando Conexões S7 .....	17
Serviços de Comunicação de "mão dupla" utilizando Conexões S7 .....	18
Configuração de Redes de Comunicação com NETPRO .....	19
Configuração de Conexões S7 .....	20
Estabelecendo Propriedades de Conexão .....	21
Compilando e Transferindo os Dados de Configuração .....	22
SFBs de Comunicação: Bloco GET (SFB 14) .....	23
SFBs de Comunicação: Bloco PUT (SFB 15) .....	24
SFBs de Comunicação: Bloco USEND (SFB 8) .....	25

## Comunicação Básica e Expandida S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.2

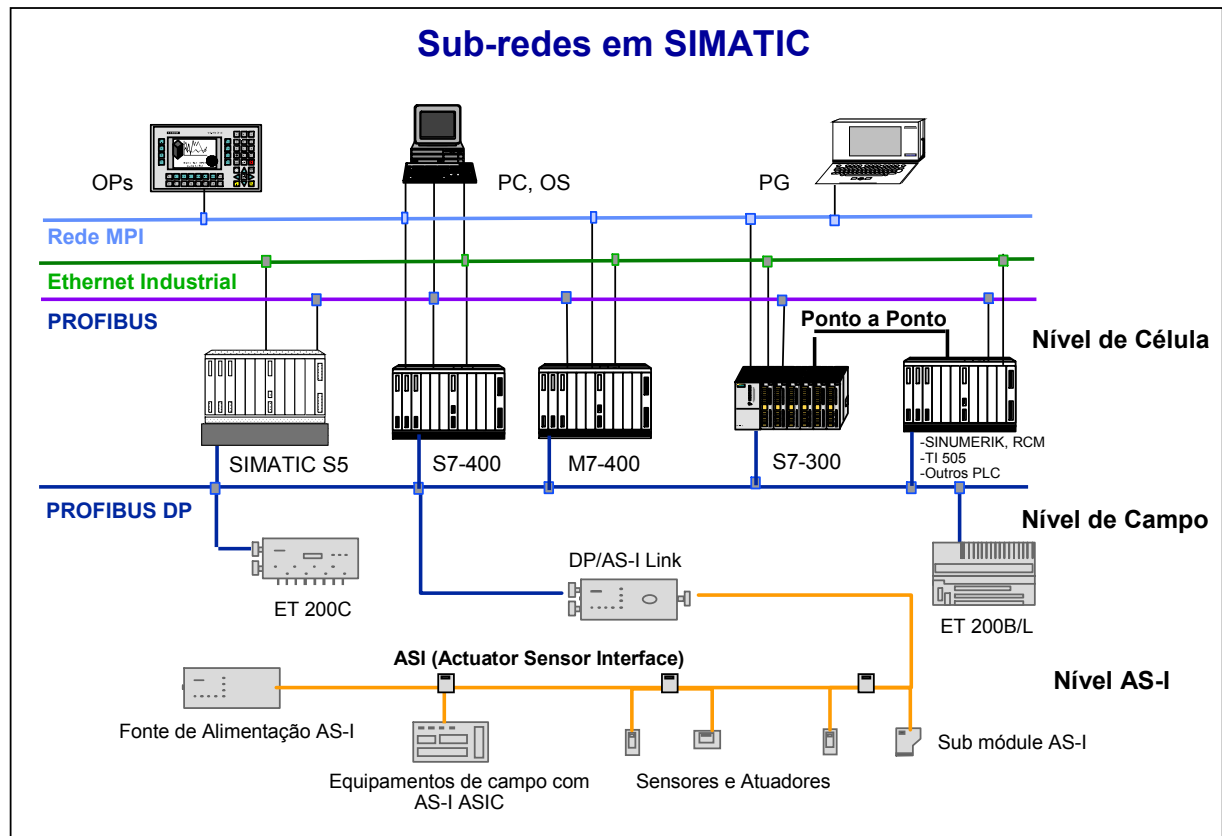


Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

SFBs de Comunicação: Bloco URCV (SFB 9) .....	26
SFBs de Comunicação: Bloco BSEND (SFB 12) .....	27
SFBs de Comunicação : Bloco BRCV (SFB 13) .....	28
SFBs de Comunicação : Bloco STOP (SFB20) .....	29
SFBs de Comunicação : Bloco START (SFB19) .....	30
SFBs de Comunicação : Bloco controle (SFC 62) .....	31
Exercício 10.1: Configurando uma conexão S7 .....	32
Exercício 10.2: Comunicação com os SFBs GET/PUT .....	33
Exercício 10.3: Comunicação com os SFBs START/STOP .....	34

**SIMATIC S7**

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.3Conhecimento em Automação  
Training Center**Vista Geral**

SIEMENS oferece as seguintes sub-redes, dependendo dos vários requisitos para as tarefas de comunicação no nível de célula (tempo não crítico) ou no nível de campo (tempo crítico).

**MPI**

A sub-rede MPI é projetada para tarefas no nível de célula. A MPI é uma interface com característica multi-point em SIMATIC S7.

Ela é projetada como interface PG, isto é, para a conexão de PGs (comissionamento e teste) e OPs (interface com operador). Entre estas opções, a sub-rede MPI pode também ser usada como rede entre poucas CPUs.

**Ethernet Industrial**

Ethernet Industrial é um sistema de comunicação SIMATIC em protocolo aberto para o nível de gerenciamento e o nível de célula.

Ethernet Industrial is projetada para transmissão de grande quantidade de dados com tempo não crítico e oferece a possibilidade de conexão com outras redes de comunicação através de Gateways.

**PROFIBUS**

PROFIBUS é um sistema de comunicação SIMATIC em protocolo aberto para o nível de célula e o nível de campo. Existem duas versões, cada uma com suas características próprias:

- no nível de célula como PROFIBUS para comunicação em tempo não crítico entre nós igualmente inteligentes.
- como comunicação de campo PROFIBUS DP para tempo crítico, troca de dados cíclicos entre mestres inteligentes e equipamentos de campo.

**Conexão PtP**

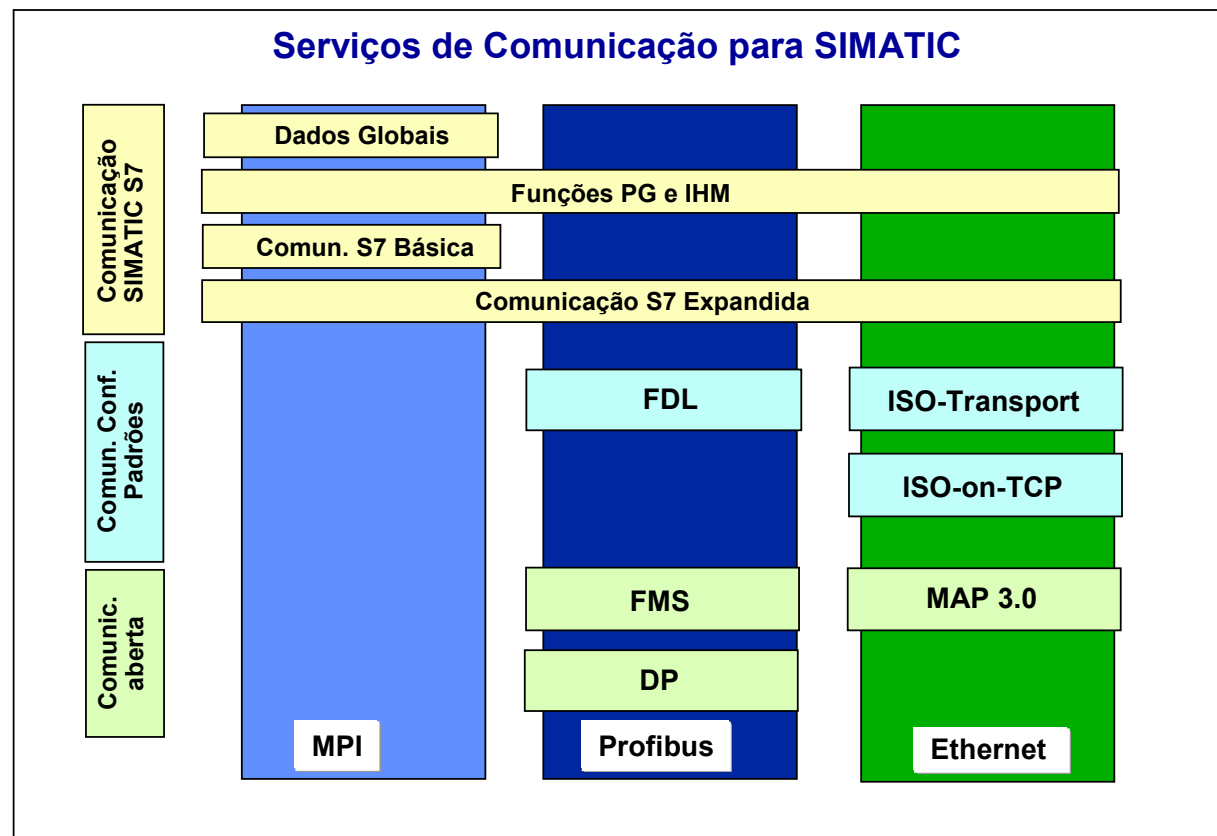
Conexões Ponto a Ponto são principalmente usadas para troca de dados com tempo não crítico entre duas estações ou para a conexão de equipamentos, como OPs, impressoras, leitoras de códigos de barra, leitoras de cartões magnéticos etc. para uma estação.

**AS Interface**

A Interface Sensor Atuador é uma sub-rede para o nível mais baixo de processo sistemas de PLC. Com sua ajuda, sensores binários e atuadores podem ser ligados em rede de comunicação.



## Serviços de Comunicação para SIMATIC



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.4



Conhecimento em Automação  
Training Center

#### Serviços

Um serviço de comunicações descreve funções de comunicações com características de performance definidas, como troca de dados, equipamentos de controle, equipamentos de monitoração e transferência de programas.

#### Dados Globais

GD (Global Date – Dados Globais) para troca de dados cíclicos de pequenas quantidades de dados (em S7-400 adicionalmente mais dados).

#### Comunicação S7

Esta ferramenta de comunicação é otimizada para a comunicação de PLCs S7, PGs/PCs e OP/TDs nas conexões SIMATIC S7.

- Funções PG; uma PG pode ser conectada sem configuração de conexão.
- Funções IHM; um OP pode ser conectado sem configuração de conexão.
- Comunicação Básica é implementada com SFCs, estas estão inseridas no sistema operacional das CPUs. (comunicação SFC são executadas sem configuração de conexão).
- Comunicação Expandida ocorre através de conexões configuradas com a ajuda de SFBs (S7-400 Client/Server; S7-300 somente Server).

#### FDL (SDA)

Para a transferência segura de dados de uma quantidade média de dados entre SIMATIC S7 e S5. Corresponde ao Layer 2 *Fieldbus Date Link (FDL)* para Profibus.

#### ISO Transport

É utilizado para a transferência segura de dados entre SIMATIC S5 e S7. Usado para transferir quantidade média de dados (até 240 bytes).

#### ISO-on-TCP

É utilizado para a transferência segura de dados de uma quantidade média de dados do SIMATIC S7 para PCs ou sistemas não Siemens através de rede TCP/IP. As ferramentas FDL, ISO e ISO-on-TCP são disponibilizadas através da chamada função AG-SEND/AG-RECEIVE.

#### FMS

*Fieldbus Message Specification (FMS)* faz comunicação orientada a objeto entre parceiros inteligentes bem como também com equipamentos de campo. Utilitários suportados por FMS (variáveis, serviços de domínio, etc.) são especificados na EN 50170 Vol. 2.

#### MAP

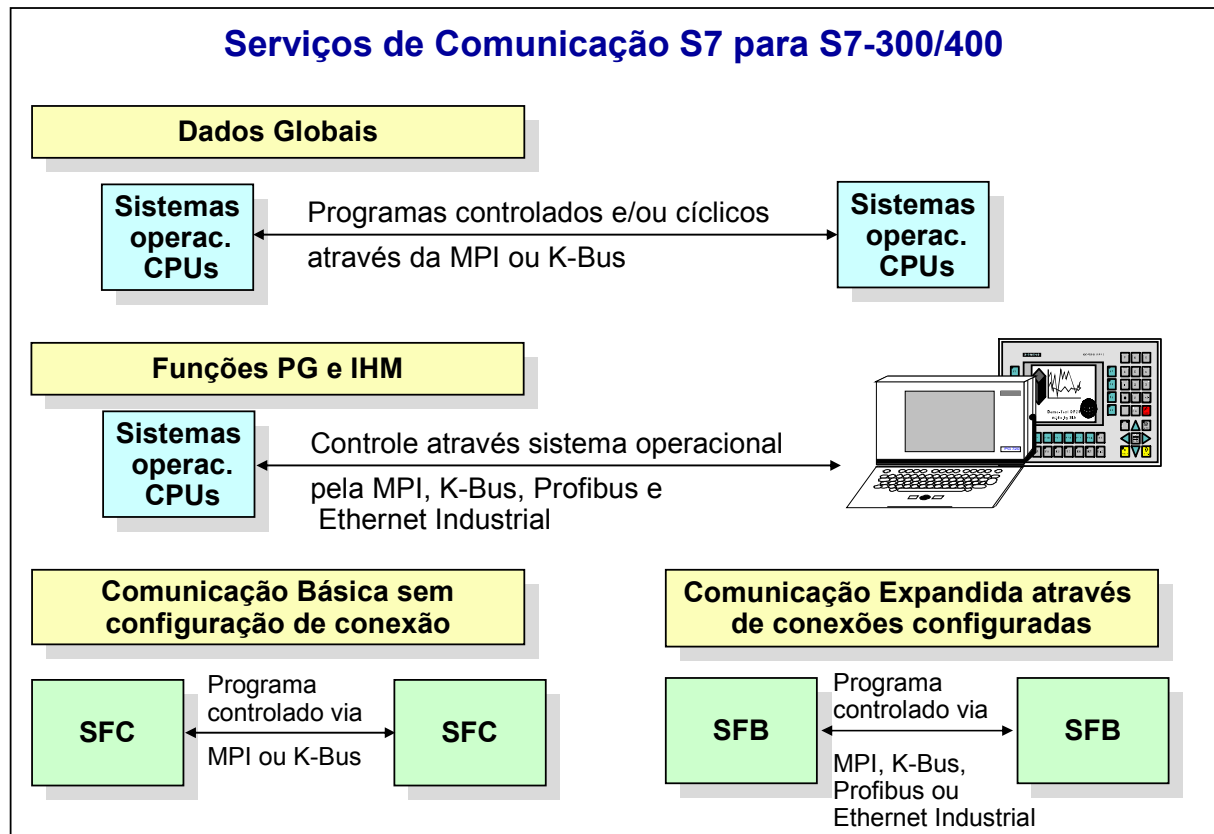
Originalmente desenvolvido pela indústria automobilística americana General Motors, este protocolo é para comunicação orientada a objeto entre sistemas PLC (MAP= *Manufacturer Automation protocolo*).

#### DP

O protocolo DP (I/O Distribuído) é especialmente otimizado para comunicação orientada a objeto com tempo crítico de unidades de controle inteligentes (DP Mestres) para equipamentos de campo (EN 50170 Vol. 3).



## Serviços de Comunicação S7 para S7-300/400



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.5



Conhecimento em Automação  
Training Center

#### Dados Globais

Esta comunicação é aplicável para troca de dados cíclicos entre CPUs utilizando a interface MPI e sem um programa. A troca de dados acontece em um ponto de controle do ciclo, junto com a atualização da imagem de processo.

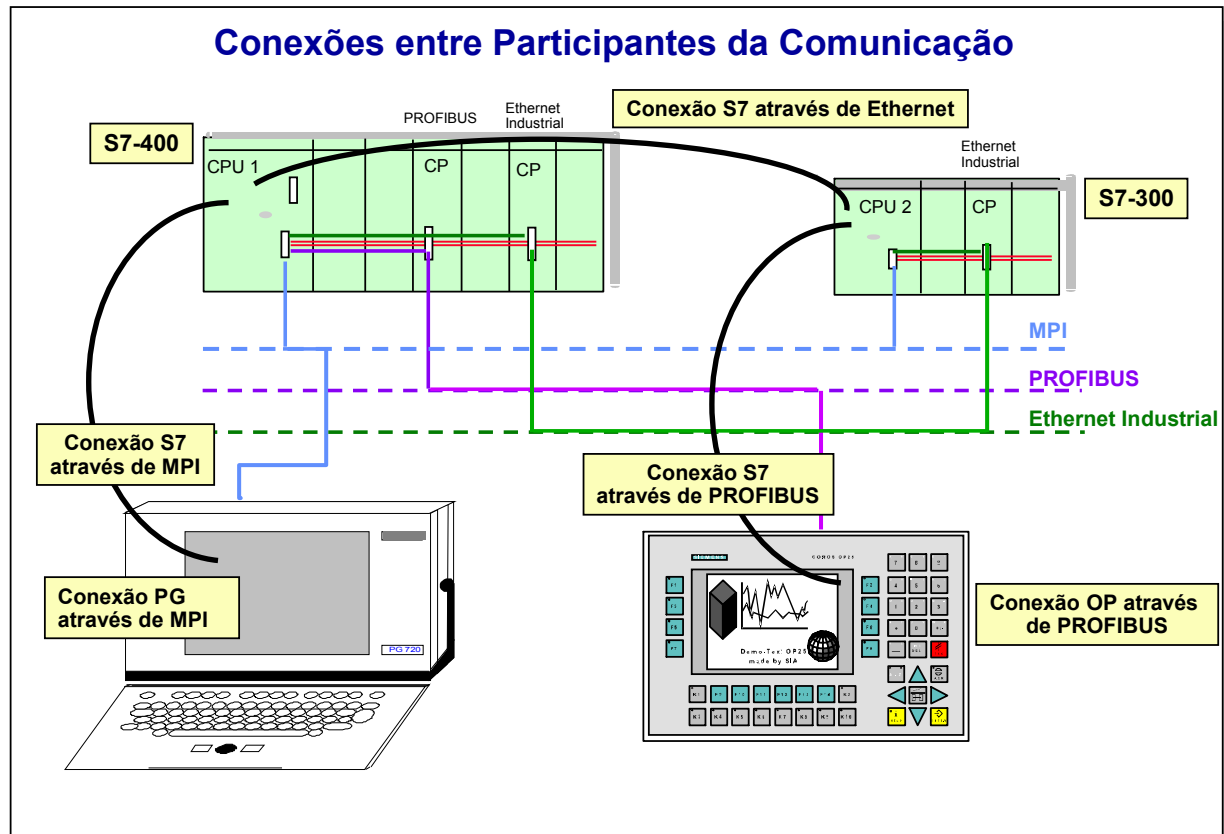
#### Função PG e IHM

Serviços do sistema como funções PG e IHM são baseadas, em análise final, na comunicação S7 Expandida. Pré requisito para a conexão de um PG ou uma IHM a um sistema S7-300/400 é a sua disponibilidade de uma conexão livre à conexão com o parceiro (S7-CPU, M7-CPU, M7-FM, etc.).

**Comunicação Básica** Com este serviço de comunicação, dados para todas as CPUs S7-300/400 podem ser transferidos por meio da sub-rede MPI ou com a estação através do K bus. Funções do Sistema (SFCs), como a X\_SEND pelo lado de envio e X\_RCV pelo lado de recepção, são chamados no programa do usuário. A quantidade de dados do usuário que podem ser transferidas em uma chamada é de um máximo de 76 bytes. A conexão da comunicação ao parceiro é ativamente configurada quando os SFCs são chamados e desconectada após a transmissão. Uma configuração de conexão não é necessária para isto.

#### Comunicação Expandida

Você pode usar este serviço de comunicação para todas as CPUs S7-400. A quantidade máxima de 64KBytes de dados pode ser transferida por meio de várias sub-redes (MPI, K-Bus, Profibus e Ethernet Industrial). Blocos de função do sistema (SFBs) são utilizados como interface de programação. Estes SFBs somente são integrados no sistema operacional das CPUs S7-400, eles não existem no S7-300. Ao lado da função de transmissão dados, este serviço de comunicação também contém funções de controle como START e STOP do PLC parceiro. A comunicação é implementada através da configuração de conexões (tabela de conexão). Estas conexões são configuradas durante a energização da estação e permanentemente continuará a existir.



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.6



Conhecimento em Automação  
Training Center

## Conexões

Uma conexão é uma atribuição lógica de dois parceiros de comunicação para execução de serviços de comunicação. A conexão é diretamente ligada com o serviço de comunicação.

Cada conexão tem duas posições (em cada CPU em questão para conexões S7 ou nas CPs para conexões FDL), as quais contêm a informação necessária para endereçamento do parceiro da comunicação bem como os atributos adicionais para a configuração da conexão.

Conexões podem ocupar um ou mais recursos de conexão nos módulos de comunicação participantes (CPUs, CPs, FMs) para cada uma das posições.

De forma a garantir uma configuração de conexão ordenada, as conexões devem ser ativa em uma das pontas e passiva na outra ponta. De outra forma, a conexão não pode ser estabelecida.

## Aplicação

Dependendo da escolha das funções de comunicação, cada conexão configurada (Comunicação Expandida) ou não configurada (Comunicação Básica) são utilizadas.

## Conexão Configurada

Este tipo de conexão é configurada com STEP 7. A conexão da posição final é atribuída uma ID local que, entre outras coisas, identifica sua própria informação de endereço e a de comunicação do parceiro.

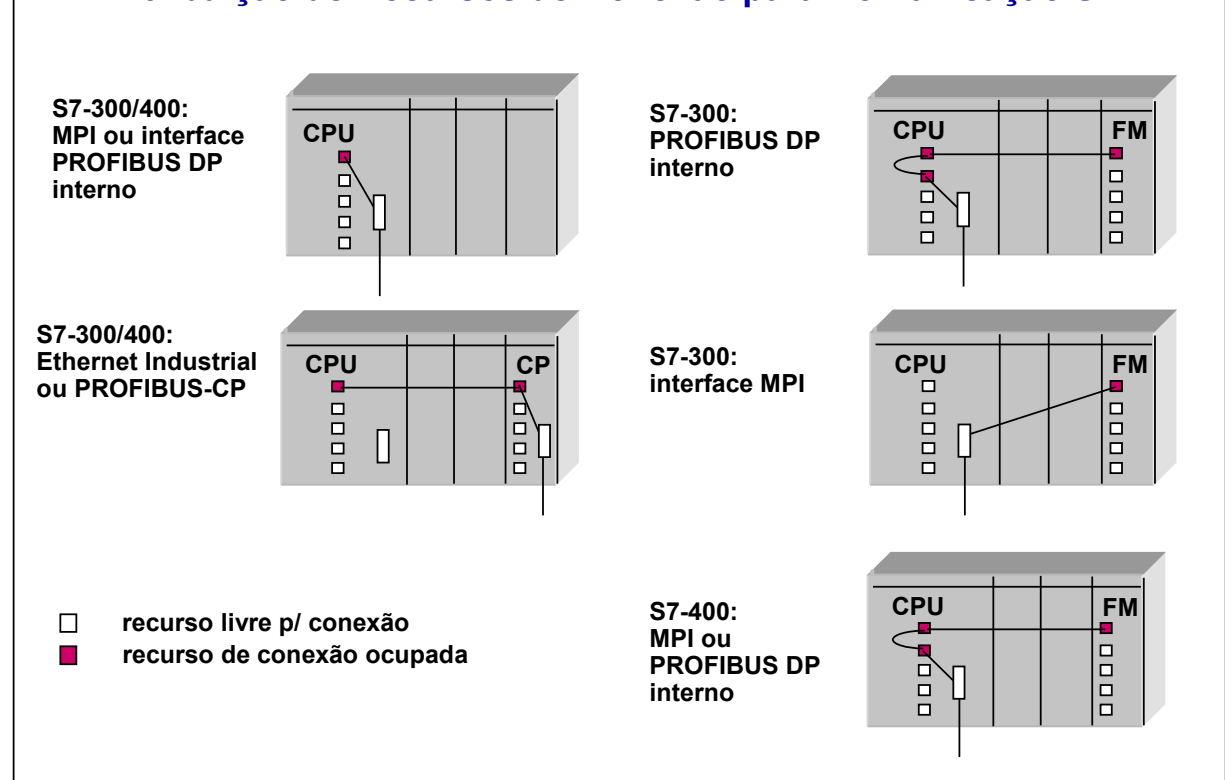
Funções de comunicação que são inicializadas por um SIMATIC OP ou PC também necessitam configurações de conexões. Estas, deste modo, são configuradas com sua própria ferramenta (p.ex. ProTool ou COML).

Configurações de conexões são configuradas por nós ativos durante energização e se mantêm configuradas durante o tempo de operação vigente.

## Conexões não configuradas

Estas conexões são configuradas quando a função de comunicação é chamada e são desconectadas após a transmissão dos dados serem completadas, se necessário.

## Atribuição de Recursos de Conexão para Comunicação S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.7Conhecimento em Automação  
Training Center

### Vista Geral

Para as estações participantes, recursos de conexão para a posição final ou para a transição de posição (p.ex. CP) são necessárias para cada conexão. O número dos recursos de conexão depende da CPU/CP.

Se todos os recursos de conexão de um parceiro de comunicação está ocupada, uma nova conexão não pode ser estabelecida.

### Funções S7 para CPUs

Para as funções S7, através da interface integrada MPI-/PROFIBUS-DP, um recurso de conexão para a posição final está ocupada por conexão S7 na CPU.

Para as funções S7 através de uma interface CP externa, cada um dos recursos de conexão está ocupado na CPU (para a posição final) e na CP (transição de posição) por conexão S7.

### Funções S7 para FMs

Para as funções S7, para um Módulo de Função (FM), através da interface interna MPI / PROFIBUS DP, dois recursos de conexão (para duas transições de posição) estão ocupadas por conexão S7 nas CPUs S7-400 e nos recursos de conexão cada uma das FMs (para a posição final) está ocupada.

Isto também é válido para cada CPU adicional (operação multi processamento) com a mesma estação, por isto as CPUs adicionais são conectadas indiretamente através do K-Bus com uma sub-rede MPI.

### PG/OPs

Cada conexão PG ou OP/TD necessita de um recurso de conexão na CPU SIMATIC S7/M7. Normalmente, um recurso de conexão para cada conexão de uma PG e um OP/TS está reservada para isto em cada CPU S7/M7.

Um recurso de conexão disponível é necessário para cada conexão PG/OP adicional. Se diversos PG/OPs estão conectados, o número de recursos de conexão disponíveis para funções S7 fica reduzido.

## Dados Característicos das CPUs S7 - Comunicação

CPU 312	IFM CPU 313	CPU 314	CPU 315/-2 DP	CPU 316	CPU 318-2
1 PG 1 OP 2 p/funç. S7	1 PG 1 OP 2 p/funç. S7 4 para SFCs	1 PG 1 OP 2 p/funç. S7 8 para SFCs	1 PG 1 OP 2 p/funç. S7 8 para SFCs	1 PG 1 OP 2 p/funç. S7 8 para SFCs	1 PG 1 OP 30 p/funç. S7 ou 30 para SFCs

CPU 412-1	CPU 413-1/2 DP	CPU 414-1/2 DP	CPU 416-1/2DP	CPU 417-4
1 PG 1 OP 14 p/funç. S7 ou 14 para SFCs	1 PG 1 OP 14 p/funç. S7 ou 14 para SFCs	1 PG 1 OP 30 p/funç. S7 ou 30 para SFCs	1 PG 1 OP 62 p/funç. S7 ou 62 para SFCs	1 PG 1 OP 62 p/funç. S7 ou 62 para SFCs

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.8Conhecimento em Automação  
Training Center

### Recursos de conexão CP

As CPs possuem os seguintes números de recursos de conexão :

#### Para CPs S7-300

CP 343-1	CP 343-1 TCP	CP 342-5	CP 343-5
16 funções S7 16 ISO-Trans	16 funções S7 16 TCP/IP	16 funções S7 16 FDL	16 funções S7 16 FDL 16 FMS

#### Para CPs S7-400

CP 443-1	CP 443-1 TCP	CP 443-5 Extended	CP 443-5 Basic
48 funções S7 64 ISO-Trans.	48 funções S7 64 TCPIP	32 funções S7 32 FDL	32 funções S7 32 FDL 32 FMS

Funções S7: para funções S7 através de PG/OPs ou SFBsISO-Trans.: conexão ISO transportTCP/IP: conexão ISO-on-TCPFDL: conexão FDLFMS: conexão FMS

## SFCs de Comunicação: Vista Geral

- Troca de dados utilizando a sub-rede MPI ou com a estação
- Nenhuma configuração de conexão necessária em comparação a comunicação com uso dos SFBs
- A conexão com o parceiro é dinamicamente configurada e desconectada
- Dados do usuário até 76 bytes
- Pode ser utilizada em todas as CPUs S7-300/400
- Variáveis também podem ser lidas e escritas no S7-200 através do PROFIBUS DP (X\_GET, X\_PUT)
- Os parceiros de comunicação também podem ser encontrados em outro projeto S7

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.9



Conhecimento em Automação  
Training Center

#### Vista Geral

Você pode trocar pequenas quantidades de dados, entre uma CPU S7/M7-300/400 e um módulo de comunicação adicional, com os SFCs de comunicação para conexões não configuradas.

Os parceiros de comunicação devem individualmente ser conectados na mesma sub-rede MPI ou serem acessados com a mesma estação através do K-Bus ou PROFIBUS DP.

Uma conexão configurada não é necessária.

#### Conexão

Quando um SFC de comunicação é chamado, uma conexão está dinamicamente configurada para o parceiro de comunicação endereçado e após completar a transmissão, dependendo do parâmetro atribuído (parâmetro: CONT) é desconectada. Para a conexão configurada, é necessário um recurso de conexão disponível para cada parceiro de comunicação.

Se em uma chamada de SFC, nenhum recurso de conexão está disponível, então um número de erro correspondente é retornado para o usuário em RET\_VAL.

Já existindo conexões, as SFBs de comunicações não podem ser utilizadas. Se a CPU ativa vai para o estado de Stop durante uma transmissão de dados, as conexões existentes são desconectadas.

As SFCs de comunicação não devem ser apagadas em modo RUN, uma vez que os recursos de conexão estejam ocupados devam possivelmente não ser habilitadas (mudanças de programa somente em estado STOP).

#### Tamanho dos

A quantidade de dados transmissíveis de usuário é de um máximo de 76 bytes

#### Dados do usuário

para todas as CPUs S7/M7/C7.

## SFCs de Comunicação: Vista Geral dos Blocos

SFC	NAME	Short Description
SFC 65	X_SEND	Bloco Send para envio de dados para o bloco X_RCV (Client)
SFC 66	X_RCV	Bloco Receive para recebimento de dados p/o bloco X_SEND
SFC 67	X_GET	Lê dados do PLC parceiro
SFC 68	X_PUT	Escreve dados do PLC parceiro
SFC 69	X_ABORT	Aborta conexão existente
SFC 72	I_GET	Lê dados da CPU parceira
SFC 73	I_PUT	Escreve dados da CPU parceira
SFC 74	I_ABORT	Aborta conexão da CPU parceira

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.10Conhecimento em Automação  
Training Center

### Vista Geral

As SFCs de comunicação oferecem a possibilidade de uma transmissão de dados reconhecidos utilizando conexões S7 não configuradas.

Com as SFCs de comunicação (X\_...), você pode endereçar todos os parceiros de comunicação na mesma sub-rede MPI, com as SFCs (I\_...) todos os parceiros de comunicação com um endereço de I/O (p.ex. FMs, etc.) com a mesma estação.

Comunicação utilizando uma sub-rede MPI então também é possível, se o parceiro de comunicação é encontrado em outro projeto S7.

O número de nós de comunicações sucessivamente acessíveis não é limitado.

### Endereçamento

Em comunicação (X\_...) utilizando uma sub-rede MPI, o endereçamento do parceiro toma lugar pela especificação do endereço MPI, em comunicação (I\_...) com a mesma estação, pela especificação do endereço lógico inicial do módulo (endereço I/O).

Se um módulo tem um endereço base para entradas (endereços I) bem como um para saídas (endereços Q), então em uma chamada de SFC, o menor dos dois deve ser pego.

### Consistência de dados

O tamanho máximo da área de dados que pode ser lida (X\_PUT, I\_PUT) e escrita (X\_GET, I\_GET) como um bloco relacionado pelo sistema operacional com CPUs S7-300/400, é designado como consistência de dados.

Com S7-300/400, a consistência de dados é:

- CPUs S7-300: 8 Bytes
- CPUs S7-400: 32 Bytes

Então, por exemplo, um array de dados tipo byte, word ou double word pode ser transmitido consistentemente até o tamanho máximo.

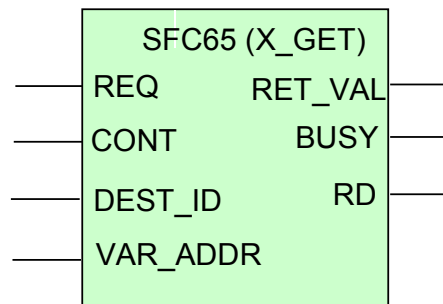
## SFCs de Comunicação: Bloco X\_GET (SFC 67)

### Representação STL

Exemplo com parâmetros atribuídos

```
CALL SFC 67
REQ:= I 0.4           //Gatilho
CONT:= FALSE          //Disc. conexão
DEST_ID:= W#16#3      //Endereço MPI
VAR_ADDR:= P#M20.0 BYTE 10 //Var. Remota
RET_VAL:= MW100       //Código de erro
BUSY:= M 4.1          //SFC ativa
SD:= P#M0.0 BYTE 10  //Variável Local
```

### Representação LAD/FBD



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.11



Conhecimento em Automação  
Training Center

### Descrição

Com a SFC 67 (X\_GET), você pode ler dados de um parceiro de comunicação que não está na estação S7 local. Não existe SFC correspondente no parceiro de comunicação.

O job leitura é ativado após a chamada do SFC com REQ=1. Depois disto, você continua a chamar a SFC até que a recepção do dado esteja indicada pelo BUSY=0. RET\_VAL então contém o comprimento do bloco de dados recebido em bytes.

Assegure-se que a área de recepção definida pelo parâmetro RD (na CPU de recepção), seja pelo menos tão longa quanto a área definida a ser lida pelo parâmetro VAR\_ADDR (no parceiro de comunicação). O tipo de dado RD e VAR\_ADDR devem também combinar.

Parâmetros da SFC 67 X\_GET

Parâmetros	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L,const.)	Ativa uma transferência com sinal 1
CONT	INPUT	BOOL (I,Q,M,D,L,const.)	CONT=0 desconecta conexão CONT=1 conecta remanescente
DEST_ID	INPUT	WORD (I,Q,M,D,L, const.)	Endereço MPI do parceiro
VAR_ADDR	INPUT	ANY (I,Q,M,D)	Referência para área (CPU remota), de onde os dados são lidos
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Retorno do valor com o código de erro
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 Função envio operando BUSY=0 Função envio completada
RD	OUTPUT	ANY (I,Q,M,D,L)	Referência para área (CPU local), na qual os dados lidos são escritos

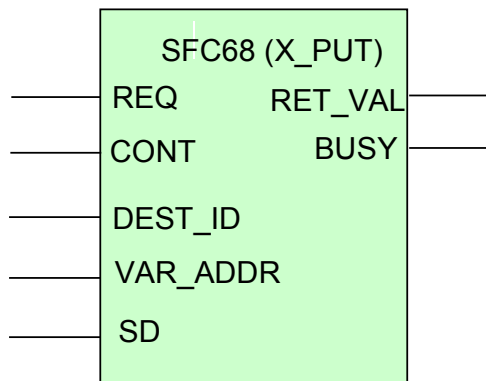
## SFCs de Comunicação: Bloco X\_PUT (SFC 68)

### Representação STL

Exemplo com parâmetros atribuídos

```
CALL SFC 68
REQ:= I 0.5           //Gatilho
CONT:= FALSE          //Disc. conexão
DEST_ID:= W#16#3      //Endereço MPI
VAR_ADDR:= P#M20.0 BYTE 10 //Var. Remota
SD:= P#M0.0 BYTE 10  //Variável Local
RET_VAL:= MW100       //Código de erro
BUSY:= M 4.1         //SFC ativo
```

### Representação LAD/FBD



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.12



Conhecimento em Automação  
Training Center

### Descrição

Com a SFC 68 (X\_PUT), você pode ler dados de um parceiro de comunicação que não está na estação S7 local. Não existe SFC correspondente no parceiro de comunicação.

O job leitura é ativado após a chamada do SFC com REQ=1. Depois disto, você continua a chamar a SFC até que o reconhecimento seja recebido com BUSY=0.

Assegure-se que a área de envio definida pelo parâmetro RD (na CPU de envio), seja do mesmo tamanho que a área definida a ser lida pelo parâmetro VAR\_ADDR (no parceiro de comunicação). O tipo de dado SD e VAR\_ADDR devem também combinar.

Parâmetros da SFC 68 X\_PUT

Parâmetros	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L, Const.)	Ativa uma transferência com sinal 1
CONT	INPUT	BOOL (I,Q,M,D,L, Const.)	CONT=0 desconecta conexão CONT=1 conecta remanescente
DEST_ID	INPUT	WORD (I,Q,M,D,L, Const.)	Endereço MPI do parceiro
VAR_ADDR	INPUT	ANY (I,Q,M,D)	Referência para área (CPU remota) na qual está escrevendo
SD	INPUT	ANY (I,Q,M,D)	Referência para área (CPU local), que contem os dados a serem transferidos
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Retorno do valor com o código de erro
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 Função envio operando BUSY=0 Função envio completada



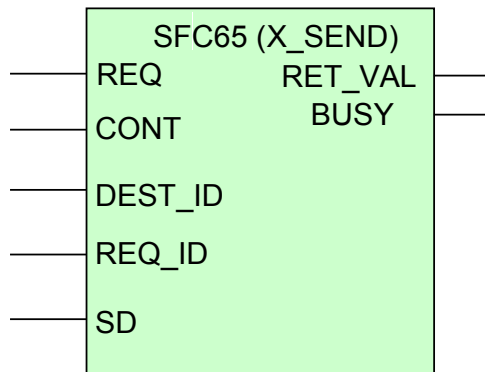
## SFCs de Comunicação: Bloco X\_SEND (SFC 65)

### Representação STL

Exemplo com parâmetros atribuídos

```
CALL SFC 65
REQ:= M4.0           //Gatilho
CONT:= FALSE         //Disc. conexão
DEST_ID:= W#16#4     //Ender. MPI
REQ_ID:= DW#16#1     //Identificador
SD:= P#M20.0 BYTE 10 //Variável
RET_VAL:= MW40       //Código de erro
BUSY:= M 4.1        //SFC activa
```

### Representação LAD/FBD



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.13



Conhecimento em Automação  
Training Center

### Descrição

Com a SFC 65 (X\_SEND), você pode enviar dados de um parceiro de comunicação que está do lado de fora da estação S7 atual. Dados recebidos no parceiro de comunicação ocorre através da SFC 66 (X\_RCV).

Você pode identificar seu dado enviado com o parâmetro de entrada REQ\_ID. Este identificador de job também é transmitido. Você pode disponibilizá-lo ao parceiro de comunicação, de forma a determinar a origem do dado.

A função de envio ocorre após a chamada da SFC com REQ=1.

Você deve se assegurar que o envio da área (na CPU que envia) definida através the parâmetro SD is smaller ou the mesma as the receive área (at the comunicação parceiro) definida através the parâmetro RD.

Parâmetros da SFC 65 X\_SEND

Parâmetros	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L, Const.)	Ativa uma transferência com sinal 1
CONT	INPUT	BOOL (I,Q,M,D,L,Const.)	CONT=0 desconecta conexão CONT=1 conecta remanescente
DEST_ID	INPUT	WORD (I,Q,M,D,L, Const.)	Endereço MPI do parceiro
REQ_ID	INPUT	DWORD (I,Q,M,D,L, const.)	Solicita um ID para identificar os dados do parceiro
SD	INPUT	ANY (I,Q,M,D)	Referência para área de envio
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Retorno do valor com o código de erro
BUSY	OUTPUT	BOOL (I,Q,M,D)	BUSY=1 Função envio operando BUSY=0 Função envio completada

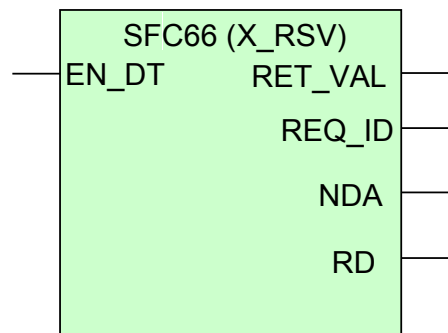
## SFCs de Comunicação: Bloco X\_RCV (SFC 66)

### Representação STL

Exemplo com parâmetros atribuídos

```
CALL SFC 66
EN_DT:= TRUE           //Gatilho trans. dados
RET_VAL:= MW 50        //Código de erro
REQ_ID:= MD52          // ID do Job
NDA:= M40.0            //Dados disponíveis
RD:= P#M20.0 BYTE 10  //Variável
```

### Representação LAD/FBD



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.14



Conhecimento em Automação  
Training Center

### Descrição

Com a SFC 66 (X\_RCV) você recebe dados que um ou mais parceiro(s) de comunicação enviam com a SFC 65 (X\_SEND). Este (estes) parceiro(s) de comunicação estão do lado de fora da estação S7 atual.

Com a SFC 66 (X\_RCV) você pode:

- Determinar se neste momento (instante) os dados enviados estão disponíveis. Estes estavam, se necessário, alocados em uma fonte interna no sistema operacional.
- Copiar o bloco de dados mais velho, que está disponível na fonte, em uma área de recebimento especificada por você.

A seleção ocorre através do parâmetro de entrada EN\_DT (habilita transferência de dados).

Parâmetros da SFC 66 X\_RCV

Parâmetros	Modo	Tipo	Significado
EN_DT	INPUT	BOOL (I,Q,M,D,L, const.)	EN_DT=0 verifica qual bloco de dado está presente EN_DT=1 copia bloco de dado para memória
RET_VAL	OUTPUT	INT (I,Q,M,D,L)	Retorna valor com o código de erro.
REQ_ID	OUTPUT	DWORD (I,Q,M,D,L)	Solicita identificador para a SFC X_SEND 66, cujos dados estão presentes na primeira posição da fonte
NDA	OUTPUT	BOOL (I,Q,M,D,L)	NDA=0 nenhum bloco de dados presente NDA=1 pelo menos um bloco de dados presente (para EN_DT=1) ou bloco de dados foi copiado para memória (EN_DT=1)
RD	OUTPUT	ANY (I,Q,M,D)	Referência para área de recebimento

## SFBs de Comunicação: Vista Geral

- Troca de dados utilizando MPI, K-Bus, Profibus ou Ethernet Industrial
- Configuração das conexões através da tabela de conexão
- As conexões são configuradas durante o restart completo e existem permanentemente (exceto no modo STOP)
- Tamanho dos dados do usuário até 64 KBytes
- Serviços de comunicações também para controle (Stop, Start) do parceiro
- SFBs existem para todas CPUs S7-400
- Dados também podem ser lidos e escritos por um S7-300 (GET/PUT)
- Diferentes tarefas podem ser manipuladas através de uma conexão

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.15



Conhecimento em Automação  
Training Center

#### Vista Geral

Os blocos SFB estão disponíveis em todas as CPUs S7-400 e são utilizados para trocar dados com as CPUs S7/M7-300/400. Dados até 64 Kbytes podem ser transferidos deste modo por várias sub-redes (MPI, Profibus, Ethernet Industrial) com estes blocos.

#### Conexões

Os SFBs de comunicações oferecem a possibilidade para uma transmissão de dados protegidos utilizando as conexões S7 configuradas. A configuração destas conexões ocorrem com a ajuda da ferramenta "Netpro" (configurador de conexões), o qual opera em conjunto com o SIMATIC Manager.

As configurações de conexões são realizadas durante o RESTART COMPLETO das estações e ficam permanentemente, exceto quando a estação vai para o modo STOP. Durante um restart, as conexões não são configuradas novamente.

comunicação is exclusively possível entre estações of an S7-Project. The comunicação parceiros must be conectada on a common MPI-, PROFIBUS- ou Industrial Ethernet sub-rede.

#### SFBs

As interfaces para comunicação S7 para o programa do usuário no SIMATIC S7 formam blocos S7 especiais do tipo SFB. Os SFBs são orientados pelo padrão ISO/IEC 1131-5 e oferecem uma interface uniforme para o usuário.

As conexões devem ser configuradas para a comunicação. Os números de conexão referenciam a atribuição de nós e o meio de transmissão por meio dos Números de Identificação. Estes Números de Identificação são tratados como parâmetro de bloco "ID" durante uma chamada de SFB.

#### Dados do Usuário

O tamanho dos dados do usuário depende do bloco utilizado e do parceiro de comunicação :

- PUT/GET 160 bytes para o S7-300 e 400 bytes para o S7-400/M7
- USEND/UREC até 440 bytes
- BSEND/BRCV até 64KBytes

## SFBs de Comunicação: Vista Geral dos Blocos

SFB/SFC	NOME	Tipo de Com.	Descrição abreviada
SFB 8	USEND	mão dupla	Bloco de Envio p/envio de dados ao bloco URCV (Client)
SFB 9	URCV	mão dupla	Bloco de Recebimento p/recebimento de dados do bloco USEND
SFB 12	BSEND	mão dupla	Bloco de Envio p/envio de grandes blocos de dados p/ bloco BRCV (até 64 KByte)
SFB 13	BRCV	mão dupla	Bloco de Recebimento p/recebimento de grandes blocos de dados (até 64 Kbyte)
SFB 14	GET	mão única	Leitura de dados do PLC parceiro
SFB 15	PUT	mão única	Escrita de dados para o PLC parceiro
SFB 16	PRINT	mão única	Envio de dados para impressora remota
SFB 19	START	mão única	Realiza restart completo no parceiro
SFB 20	STOP	mão única	Coloca o parceiro em modo Stop
SFB 21	RESUME	mão única	Realiza restart no parceiro
SFB 22	STATUS	mão única	Verifica o estado do parceiro (RUN, STOP, start-up, hold)
SFB 23	USTATUS	mão única	Recebe mensagens de estado do parceiro
SFC 62	CONTROL	---	verifica o estado interno de uma conexão S7 + SFB

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.16Conhecimento em Automação  
Training Center

#### SFBs: S7- 400

Os SFBs para comunicação S7 estão integradas como blocos de função do sistema (SFBs) no sistema operacional das CPU's S7-400.

Para integração ao programa do usuário, o usuário pode localizar os cabeçalhos de blocos na *Standard Library V3.x* na pasta de programa S7 *System Function Blocks*.

#### SFBs: S7 - 300

O S7-300 não dispõe de SFBs para comunicação expandida. Deste modo, o sistema operacional das CPUs S7-300 suportam a funcionalidade server dos serviços de comunicação S7 de mão única. Então, por exemplo, dados de uma CPU 3xx podem ser lidos ou escritos por uma CPU 4xx com a ajuda dos blocos GET e PUT.

#### Classes de Função

Os blocos podem ser subdivididos em um total de 4 função classes:

- Funções envio e recepção
- Funções de controle
- Funções de monitoração
- Função de verificação

#### SFBs para

##### troca de dados

Os SFBs para troca de dados são utilizados para comunicação de dados entre dois parceiros com capacidade de comunicação (S7/M7-CPU's, M7-FMs):

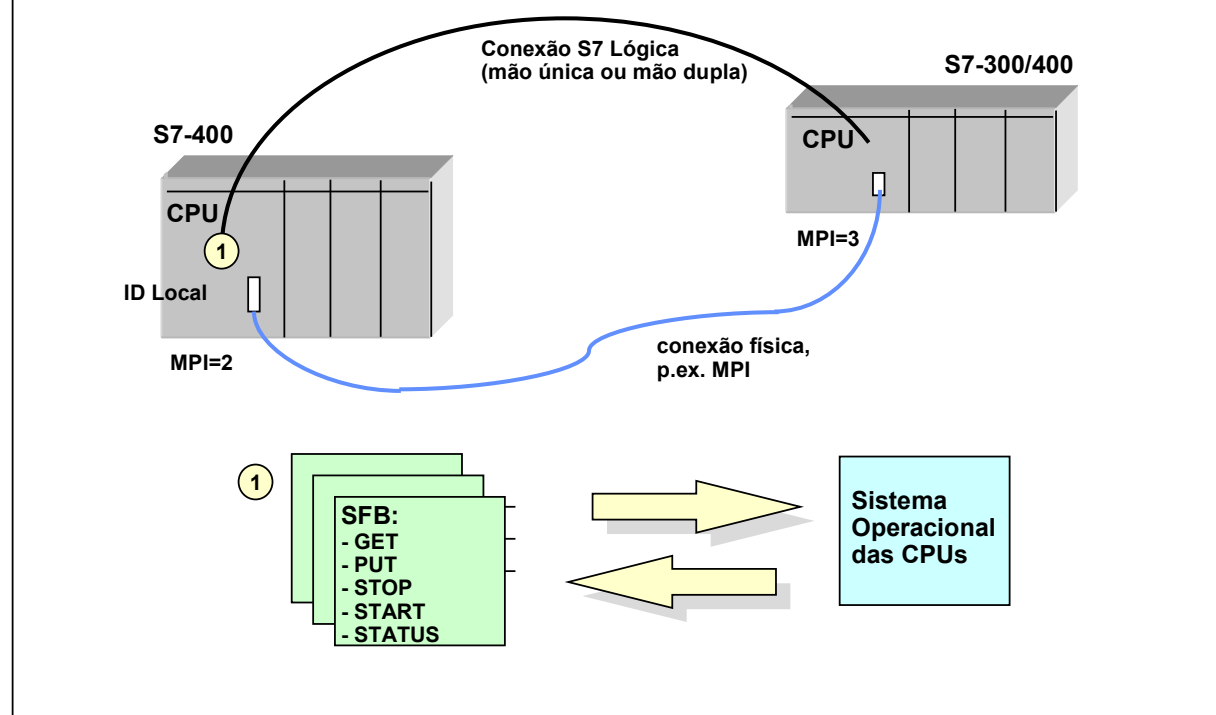
- GET, PUT (leitura e escrita de variáveis em mão única)
- USEND/URCV (mão dupla, envio/recepção sem coordenação)
- BSEND/BRCV (mão dupla, envio/recepção de blocos)

#### SFBs para gerenciamento de programas

Os SFBs para gerenciamento de programas são utilizados para o controle e avaliação do estado operacional do equipamento parceiro ou das conexões.

- START/STOP/RESUME (funções de controle)
- STATUS/USTATUS (funções de monitoração)
- CONTROL (função de verificação)

## Serviços de Comunicação de “mão única” utilizando Conexões S7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.17Conhecimento em Automação  
Training Center

### Vista Geral

Então os SFBs, nos respectivos parceiros de comunicação, podem comunicar com cada outro, as conexões S7 devem antes de mais nada serem configuradas.

As conexões S7 podem ser configuradas para redes de comunicação MPI, Ethernet Industrial e PROFIBUS.

### Conexões S7 de mão única

De uma S7-400 para uma S7-300, as conexões S7 de mão única são automaticamente ajustadas pela ferramenta de configuração. Para conexões de mão única, uma conexão ID local para identificação da conexão, isto é, o parceiro de comunicação e o meio de transmissão, estão atribuídos somente no lado da S7-400 (lado Client).

Nenhum ID de conexão é atribuído no lado da S7-300, desde que os SFBs para endereçamento da conexão de comunicação não estão localizadas no sistema operacional na CPU S7-300.

Somente os serviços de comunicação de mão única podem ser chamados através de conexões de mão única. Uma chamada do SFB correspondente somente é necessária no lado do Client (S7-400) para serviços de comunicação de mão única. Nos outros parceiros de comunicações (Server), o serviço é completamente executado pelo sistema operacional. O trabalho de programação pelo lado do usuário não é necessário no lado Server.

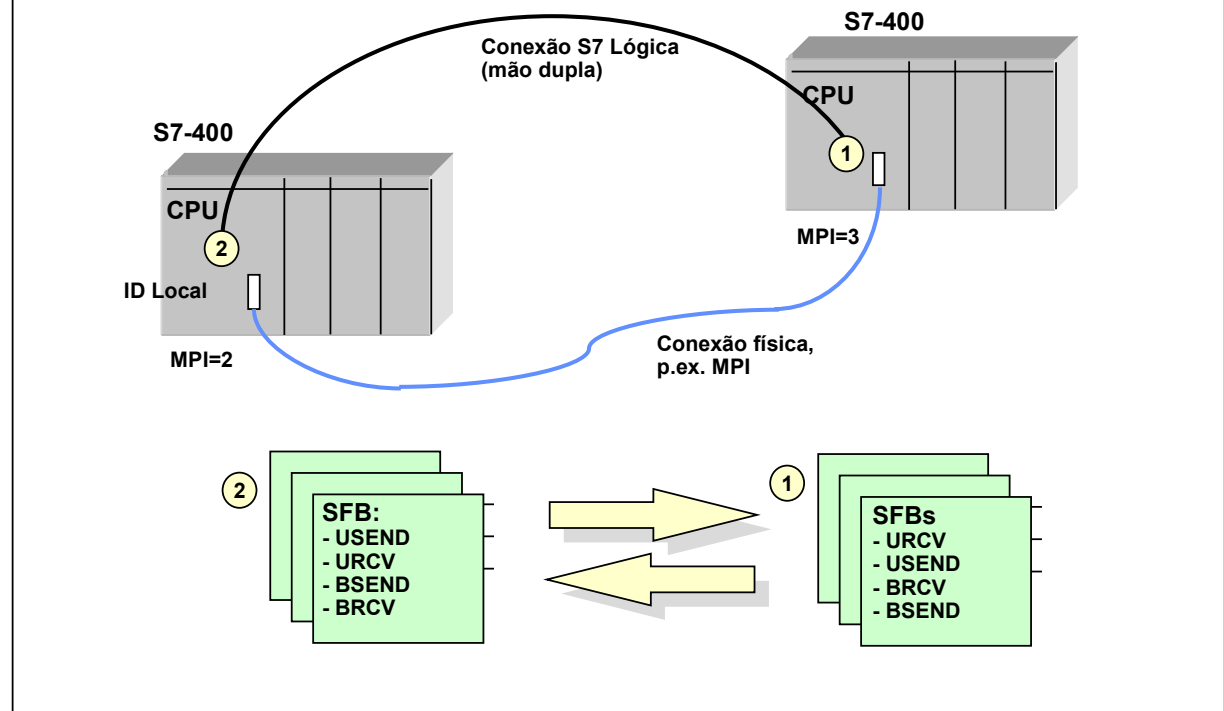
As conexões S7 de mão única são sempre configuradas pelo Client durante o start-up.

**SFBs de "mão única"** Os SFBs que são considerados de serviços de comunicação de mão única são:

- GET, PUT
- STOP, START, RESUME
- STATUS, USTATUS

Com serviços de comunicação de mão única, o programa do usuário no lado Server não é informado quando um novo dado tiver sido transmitido.

## Serviços de Comunicação de “mão dupla” utilizando Conexões S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.18



Conhecimento em Automação  
Training Center

### Conexões S7 de mão dupla

As conexões S7 de mão dupla são automaticamente ajustadas na configuração das conexões S7 entre duas CPUs S7-400. Uma ID de conexão é atribuída em cada lado da conexão de mão dupla. Ambos os lados podem então referenciar a conexão utilizando este ID de conexão.

Então, cada um dos dois parceiros pode aparecer como o “Initiator” (Client) de um serviço de comunicação.

Os serviços de comunicação de mão única (PUT, GET, etc.) bem como os de mão dupla podem ser completados utilizando conexões de mão dupla.

Com conexões S7 de mão dupla você pode decidir em qual nó configurado inicia a conexão configurada.

### SFBs de “mão dupla” Os blocos

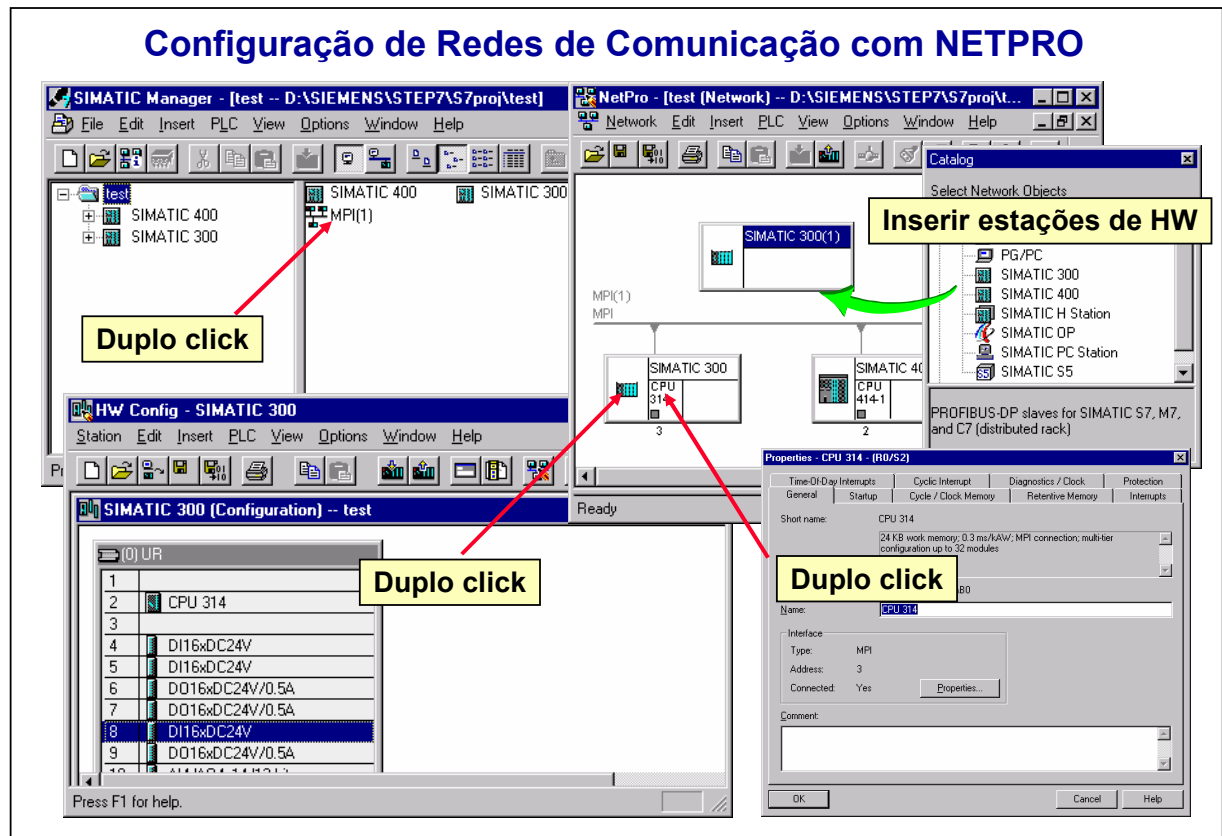
- BSEND= Envio (Client) ==> BRCV Recebimento (Server)
- USEND= Envio (Client) ==> URCV Recebimento (Server)

são considerados como SFBs de mão dupla.

Estes blocos devem sempre ser instalados em pares de blocos. As funções de comunicações de mão dupla então são sempre instaladas quando um dado transferido é utilizado para o seqüente processamento específico do dado.

Se por um lado, o receptor (Server) pode determinar pela chamada do bloco URCV ou BRCV, quando ele está pronto para receber um novo dado do transmissor para processamento seguinte. Por outro lado, o receptor pode, pela verificação dos parâmetros SFB #NDR (Novo Dado Recebido), ser informado se os novos dados foram recebidos.

## Configuração de Redes de Comunicação com NETPRO



SIMATIC S7  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.19



Conhecimento em Automação  
Training Center

### Introdução

Uma configuração gráfica de redes de comunicação (MPI, Profibus ou Ethernet Industrial) pode ser executada com a ajuda da ferramenta "NETPRO". A vantagem demonstra na clareza, documentação e a fácil chamada das ferramentas participantes como a de configuração de Hardware.

### Chamada

A chamada das ferramentas é feita com um duplo click no símbolo da rede de comunicação, por exemplo, MPI no SIMATIC Manager.

### Inserir estação HW

No catálogo você encontrará os componentes necessários como sub-redes e estações, que você pode inserir com auxílio do mouse (marca e arrasta).

### Configurando HW

Após você ter inserido as estações, você pega a ferramenta "Configure Hardware" através de duplo-clique no "símbolo de Hardware" da estação. Aqui, você pode inserir os módulos nas estações e atribuir parâmetros para eles. Para a CPU você pode, entre outras coisas, também ajustar o endereço MPI e a conexão para a sub-rede. Antes você pode configurar conexões, todas as estações participantes devem ser lincadas com a sub-rede correspondente.

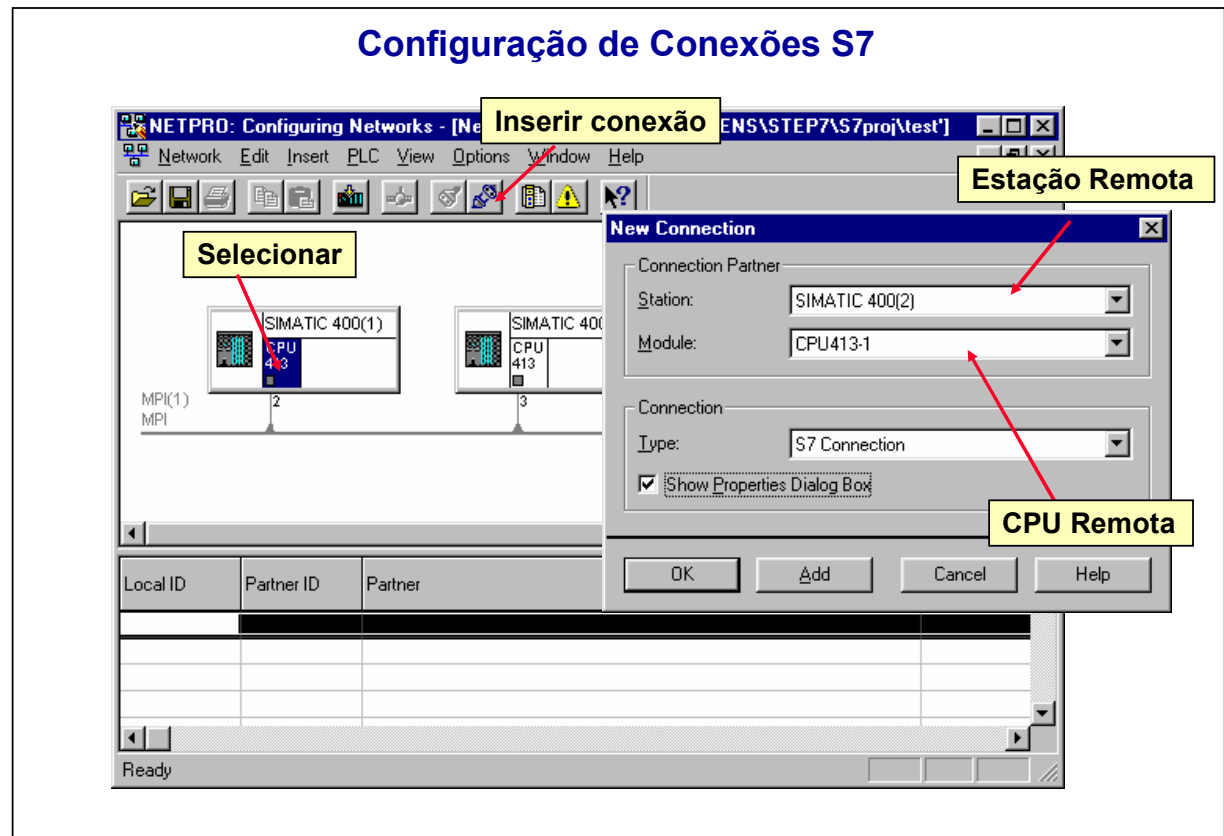
**Propriedades da CPU** Através de duplo clique na CPU com uma estação mostrada, você chega diretamente ao diálogo "propriedades- CPU". Aqui você pode ajustar as propriedades da CPU, como a para rede de comunicação, memória para osciladores, etc.

**Notar:** Quando você transfere os dados de configuração para a CPU após tê-la configurando com sucesso com o NETPRO, os parâmetros da CPU, cuja conexão à rede de comunicação, memória para osciladores etc. não são transmitidos para a CPU.

Dados de configuração da CPU, que foram modificados com NETPRO devem ser transferidos utilizando a ferramenta "HW Config"!



## Configuração de Conexões S7



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2 10P.20



Conhecimento em Automação  
Training Center

## Vista Geral

O estabelecimento das conexões de comunicação necessárias é um pré-requisito para a troca de dados controlada pelo programa utilizando SFBs. Todas as conexões que saem para fora de um módulo são mostradas na tabela de conexão pertencentes ao módulo.

## Geração de conexões

Conexões para parceiros distantes somente podem ser ajustadas, quando as estações local e distante são conectadas a mesma sub-rede.

Para inserir uma nova conexão, proceda como abaixo:

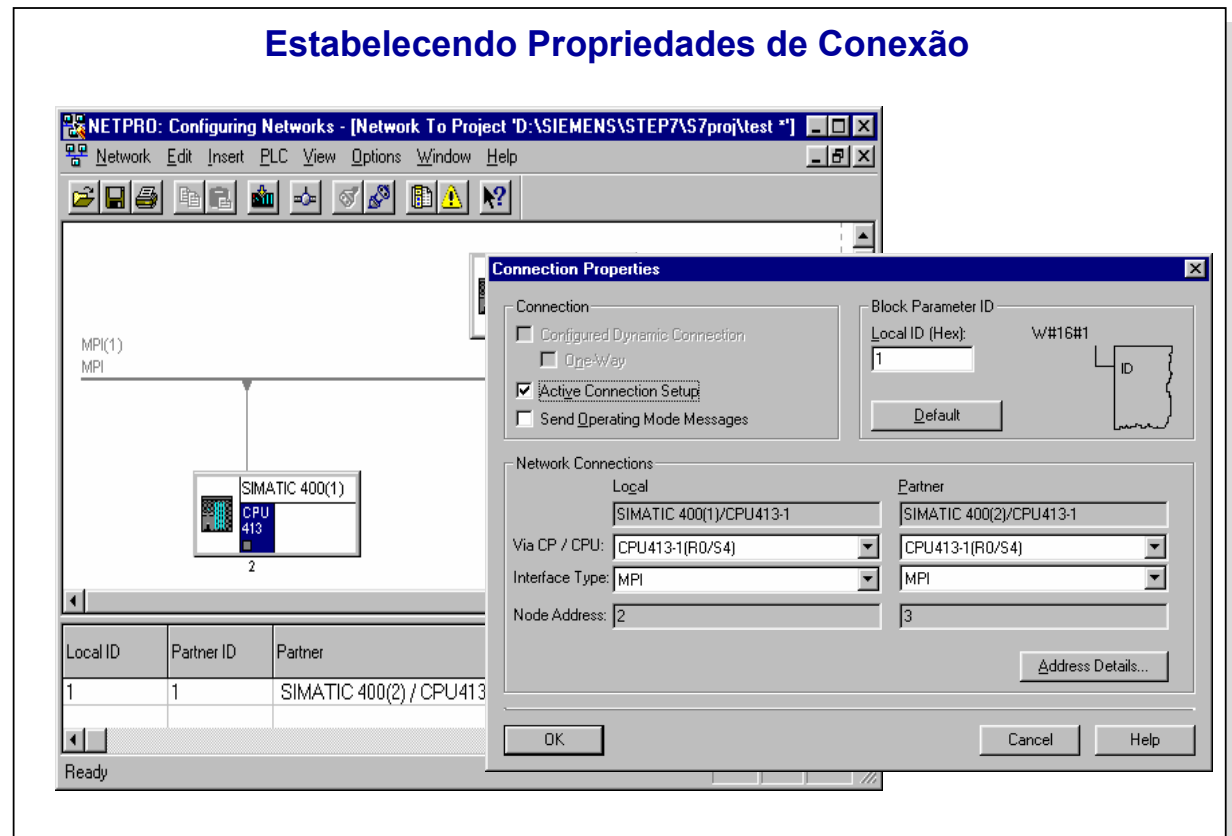
1. Nos campos "Station" e "Modul" seleciona-se os módulos programáveis, com os quais você abre uma conexão (estação local).
2. Duplo clique em uma linha vazia na tabela de conexão ou selecionando a opção de menu *Insert -> Connection...* O diálogo "New Connection" é aberto.
3. Nos campos "Station" e "Modul" seleciona-se os módulos programáveis, com os quais uma conexão é estabelecida (parceiro de conexão ou também chamada "Remote Station").
4. No campo "Type" seleciona-se o tipo de conexão: *S7 Connection*.
5. Ativar a caixa de verificação "Show propriedades diálogo Box", se após "OK" ou "Add" você deseja procurar ou mudar as propriedades de conexão.
6. Confirme suas entradas clicando no botão "OK".

**Resultado:**

O STEP 7 insere a conexão na tabela de conexões da estação local e acenta o ID Local e, se necessário, o ID do parceiro para esta conexão. Você requer estes IDs para programação dos SFBs de comunicação (valor para o parâmetro de bloco "ID").



## Estabelecendo Propriedades de Conexão



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.21



Conhecimento em Automação  
Training Center

### Vista Geral

Ao lado do estabelecimento do parceiro de conexão e do tipo de conexão, você pode, dependendo do tipo de conexão estabelecer propriedades adicionais.

### Estabelecendo Propriedades Objeto

Objetivando estabelecer propriedades especiais dos objetos da conexão de comunicação, proceder como abaixo:

1. Marcar a conexão para qual você deseja estabelecer propriedades do objeto.
2. Selecionar a opção do menu *Edit -> Object propriedades*. O diálogo "Object propriedades" é aberto.

Neste diálogo você pode estabelecer as seguintes propriedades.

### Ajustar uma conexão ativa

Você pode decidir qual dos dois nós irá prevalecer na configuração de conexão com um restart completo.

### Enviando Mensagens do Modo Operacional

Quando ativado, o nó local envia sua mensagem de modo operacional (STOP, START, HOLD,.....) para o parceiro ou para SFB 23: USTATUS para CPU parceira.

### ID Local

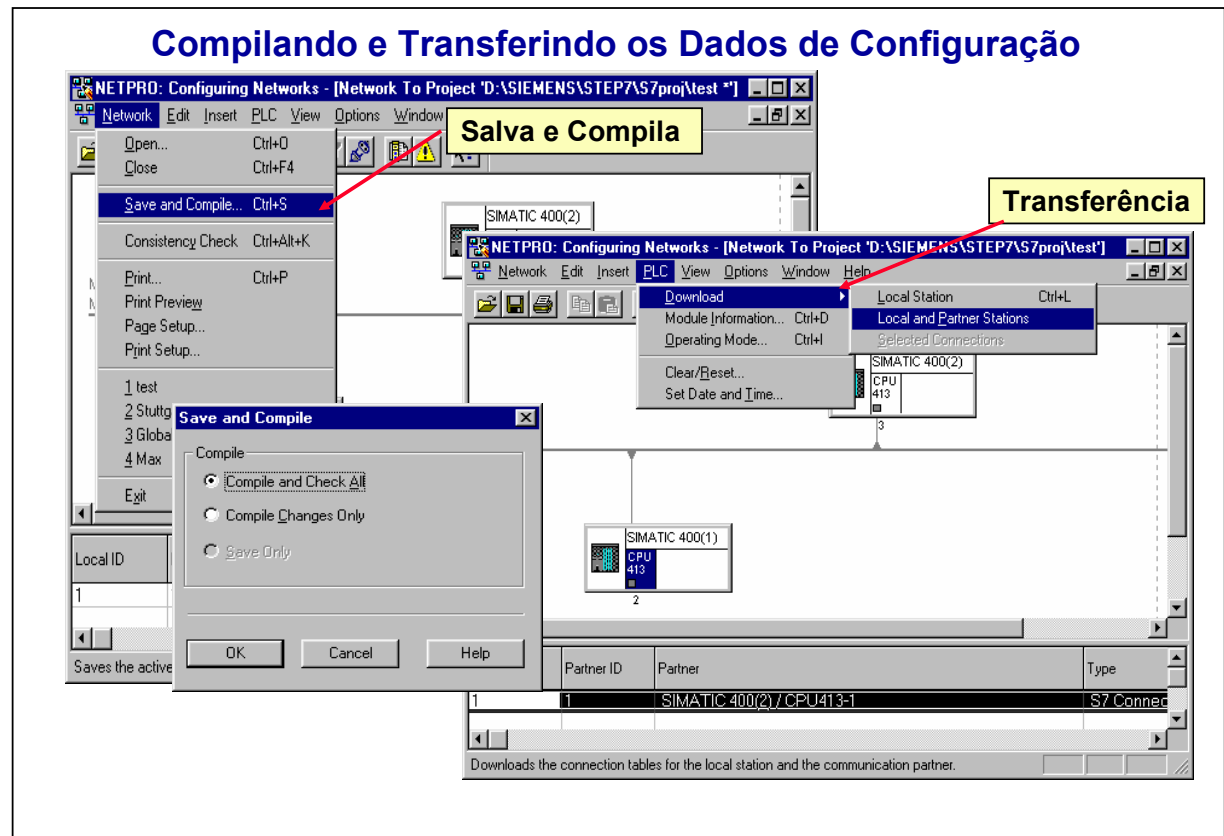
O ID local da conexão é mostrada aqui. Você pode mudar o ID local. Isto irá fazer sentido, se você já tiver programado SFBs de comunicação e você também deseja usar o ID programado na chamada para a identificação da conexão.

Você insere o novo ID local como um número hexadecimal. Ele deve ser numa faixa de valores de 1 a FFF para uma conexão S7 e não pode já ter sido atribuída.

### Conexões da Rede de Comunicação

Estes campos mostram através de qual caminho a troca de dados ocorre se diversos caminhos de comunicação (sub-redes) existem entre os dois nós, uma escolha pode ser feita através de qual caminho de comunicação a troca de dados será completada.

## Compilando e Transferindo os Dados de Configuração



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.22



Conhecimento em Automação  
Training Center

### Compilando e Salvando

Antes de você poder transferir os dados da conexão para cada estação (transferir para o PLC), a tabela de conexão deve ser salva no NETPRO e compilada com os dados de conexão. Isto ocorre com a ajuda da opção do menu *File -> Save e Compile*.

No campo de diálogo que se abre, você pode selecionar entre duas alternativas:

**Compile and Check All** : Salva todas as conexões e verifica todas as conexões para consistência com um projeto. Todas as conexões são compiladas e guardadas nos dados do sistema. No caso de inconsistência, um campo de diálogo aparece no qual os erros são mostrados.

Selecionando "Compile and Check All", se você tiver feito mudanças na configuração da rede de comunicação (p.ex. alterado endereços do nós, apagado nós ou sub-rede). É possível que conexões antigas existam e somente "Compile All and Check" dê esta informação.

**Compile Changes Only**: Salva todas as conexões do projeto e compila estas conexões que foram alteradas desde a última execução do "Save and Compile".

Quando você termina a configuração de conexão, um pergunta aparece na tela, questionando se os dados alterados estão sendo salvos ou não. Após o reconhecimento da pergunta com "Yes", os dados da conexão alterada são salvos e compilados para os dados do sistema.

### Transferindo os dados da configuração

Após salvar a tabela de conexões, os dados de conexão resultantes devem ser transferidos para os módulos participantes. A transferência da tabela de conexão para os módulos é possível através das interfaces MPI, PROFIBUS ou Ethernet Industrial do módulo.

Existem cinco caminhos para transferir os dados para os PLCs:

- Download, estação Local (menu PLC)
- Download, estações Local e parceira (menu PLC)
- Download, conexões Marcadas

(para mais informações: ver ajuda On-line)

## SFBs de Comunicação: Bloco GET (SFB 14)

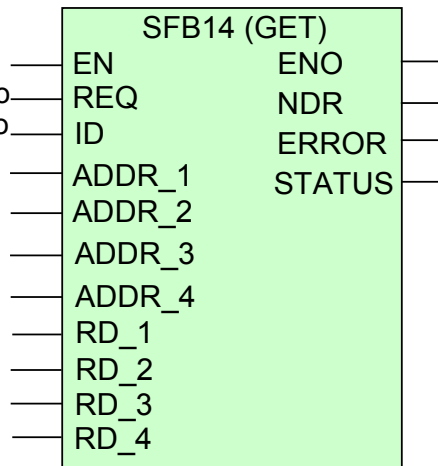
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL GET, I_GET           //Com DB Instance
REQ:=I 0.2                //Partida
ID:=W#16#1                //Número da conexão
NDR:=#NDR_FLAG            //Novo dado recebido
ERROR:= #ERROR_F          //End. com erro
STATUS:= #STATUS_W        //Inform. adicional
ADDR_1:=P#I 0.0 BYTE 1    //1. var. remota
ADDR_2:=P#I 4.0 WORD 1    //2. var. remota
ADDR_3:=                   //3. var. remota
ADDR_4:=                   //4. var. remota
RD_1:=P#Q 0.0 BYTE 1      //1. var. local
RD_2:=P#Q 4.0 WORD 1      //2. var. local
RD_3:=                    //3. var. local
RD_4:=                    //4. var. local
```

### Representação LAD

DB14 (DB Instance)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.23



Conhecimento em Automação  
Training Center

### Vista Geral

Com o SFB14 (GET) você pode ler dados de uma CPU remota.

Com uma transição positiva na entrada de controle REQ, uma tarefa de leitura é enviada à CPU parceira. O parceiro remoto retorna o dado.

Se nenhum erro ocorrer, os dados recebidos são copiados para a área configurada de recebimento (RD\_i) em uma nova chamada do SFB. O término da tarefa é indicada por um 1 no estado do parâmetro NDR.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Ativa uma transferência com uma transição positiva
ID	INPUT	WORD (I,Q,M,D,L constant)	Referencia a tabela de conexão ao número da conexão
ADDR_1 ... ADDR_4	IN_OUT	ANY (I,Q,M,D)	Ponteiro para as áreas na CPU parceira a ser lida
RD_1 ... RD_4	IN_OUT	ANY (I,Q,M,D)	Ponteiro para as área em sua própria CPU na qual os valores lidos estão sendo armazenadas. (área de dados da CPU parceira <b>ADDR_1</b> ==> <b>RD_1</b> - área de dados em sua própria CPU
NDR	OUTPUT	BOOL (I,Q,M,D,L)	Pulso de transição positivo sinaliza ao programa do usuário que existem novos dados recebidos disponíveis. "Data transferred from the partner CPU without errors."
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva sinaliza erros (pulso).
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Contem uma instrução detalhada de erro ou aviso (decimal).

## SFBs de Comunicação: Bloco PUT (SFB 15)

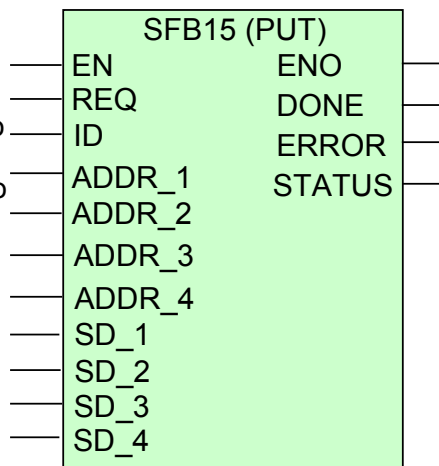
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL PUT, I_PUT(DB Instance)
REQ:=I 0.3           //Partida
ID:=W#16#1           //Número conexão
DONE:= #DONE_F        //Término c/Sucesso
ERROR:= #ERROR_F      //Término com erro
STATUS:= #STATUS_W    //Informação término
ADDR_1:=P#Q 12.0 WORD 1 //1. var. remota
ADDR_2:=              //2. var. remota
ADDR_3:=              //3. var. remota
ADDR_4:=              //4. var. remota
SD_1:=P#I 2.0 WORD 1  //1. var. local
SD_2:=               //2. var. local
SD_3:=               //3. var. local
SD_4:=               //4. var. local
```

### Representação LAD

DB15 (DB Instance)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.24



Conhecimento em Automação  
Training Center

### Vista Geral

Com o SFB15 (PUT), você pode escrever dados em uma CPU remota.

Com uma transição positiva na entrada de controle REQ, os ponteiros para as áreas a serem escritas (ADDR\_i) e os dados (SD\_i) são enviados para a CPU parceira. O parceiro remoto salva os dados necessários sobre os endereços fornecidos com os dados e retorna uma execução de reconhecimento.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Ativa uma transferência com uma transição positiva
ID	INPUT	WORD (I,Q,M,D,L constant)	Referencia a tabelade conexão ao número da conexão
ADDR_1 ... ADDR_4	IN_OUT	ANY (I,Q,M,D)	Ponteiro das áreas de dados da CPU parceira nas quais os dados da CPU que envia estão sendo escritos
SD_1 ... SD_4	IN_OUT	ANY (I,Q,M,D)	Ponteiro das áreas de dados em sua própria CPU a serem enviados à CPU parceira. (área de dados de sua própria CPU - <b>SD_1</b> ==> <b>ADDR_1</b> área de dados da CPU parceira)
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva (pulso) sinaliza o programa do usuário: transferência concluída sem erros.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva (pulso) sinaliza erros
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Contem instruções detalhadas de erros ou avisos (decimal).

## SFBs de Comunicação: Bloco USEND (SFB 8)

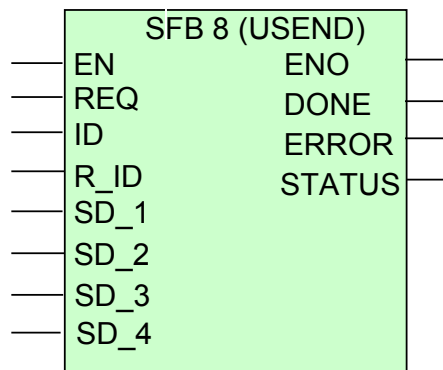
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL USEND, I_USEND(DB Instance)
REQ:= I 0.4           //Partida
ID:=W#16#3           //Número conexão
R_ID:=DW#16#B1       //Bloco par
DONE:= #DONE_F       //Término c/sucesso
ERROR:= #ERRÖR_F     //Término com erro
STATUS:= #STATUS_W   //Informação de erro
SD_1 :=P#DB3.DBX0.0 BYTE 100 //1. var. local
SD_2 :=P#DB3.DBX100.0 BYTE 100 //2. var. local
SD_3 :=P#DB3.DBX200.0 BYTE 100 //3. var. local
SD_4 :=P#DB3.DBX300.0 BYTE 154 //4. var. local
```

### Representação LAD

DB 8 (DB Instance)



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.25



Conhecimento em Automação  
Training Center

### Vista Geral

O SFB 8 (USEND) envia dados para um parceiro remoto com SFB do tipo "URCV" (o parâmetro R\_ID deve ser idêntico para ambos os SFBs). Os dados são enviados seguindo uma transição positiva na entrada de controle REQ. A função é executada sem coordenação com o SFB parceiro.

O dado a ser enviado é referenciado pelos parâmetros SD\_1 a SD\_4 mas nem todos os quatros parâmetros de envio necessitam ser utilizados.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Ativa uma transferência com uma transição positiva
ID	INPUT	WORD (I,Q,M,D,L constant)	Número da conexão do sistema S7 individual (ver tabela de conexão)
R_ID	INPUT	WORD (I,Q,M,D,L constant)	O parâmetro deve ser idêntico para ambos CFBs (USEND e URCV). Atribuição do bloco par.
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva (pulso) sinaliza o programa do usuário: transferência concluída sem erros.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva (pulso) sinaliza erros
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Mostra o estado se ERROR = 1
SD_1 ... SD_4	IN_OUT	ANY (I,Q,M,D)	Ponteiros para aquelas áreas de dados na sua própria CPU a ser enviada para CPU parceira. (área de dados de sua própria CPU SD_1 → RD_1 a área de dados da CPU parceira deve estar de acordo em número, comprimento e tipo de dado)

## SFBs de Comunicação: Bloco URCV (SFB 9)

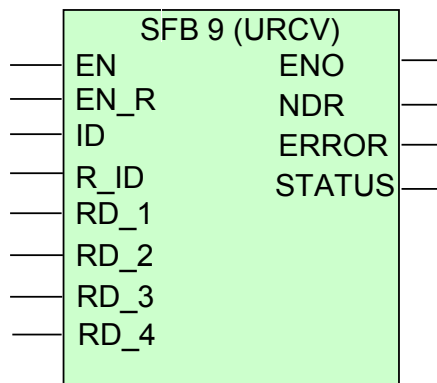
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL URCV, I_URCV           //com DB Instance
EN_R:= I 0.5                //Partida
ID:= W#16#3                 //Conexão S7
R_ID:= DW#16#B1             //Bloco par
NDR:= #NDR_F                //Novo dado recebido
ERROR:= #ERROR_F            //Término com erro
STATUS:= #STATUS_W           //Informação de erro
RD_1:=P#DB3.DBX0.0 BYTE 100 //1. var.
RD_2:=P#DB3.DBX100.0 BYTE 100 //2. var.
RD_3:=P#DB3.DBX200.0 BYTE 100 //3. var.
RD_4:=P#DB3.DBX300.0 BYTE 154 //4. var.
```

### Representação LAD

DB 9 (DB Instance)



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.26



Conhecimento em Automação  
Training Center

### Vista Geral

O SFB9 (URCV) recebe dados assíncronamente de um SFB parceiro remoto do tipo "USEND". (O parâmetro R\_ID deve se idêntico em ambos SFBs). Se o valor 1 é aplicado à entrada de controle EN\_R quando o bloco é chamado, os dados recebidos são copiados para as áreas de recepção configuradas. Estas áreas de dados são referenciadas pelos parâmetros RD\_1 a RD\_4.

Quando o bloco é primeiro chamado, a "caixa de correio de entrada" é criada. Com todas as chamadas a seguir, o dado a ser recebido deve ser acentado na caixa de correio de entrada.

Parâmetro	Modo	Tipo	Significado
EN_R	INPUT	BOOL (I,Q,M,D,L constant)	ParaRLO = 1 os dados recebidos são copiados para a área de dados configurada
ID	INPUT	WORD (I,Q,M,D,L constant)	Número da conexão para a conexão do sistema S7 individual (ver tabela de conexão)
R_ID	INPUT	DWORD (I,Q,M,D,L constant)	Os parâmetros devem ser idênticos para ambos CFBs (USEND e URCV). Atribuições dos blocos pares
NDR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva (pulso) sinaliza o programa do usuário: novos dados transferidos.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva = erro (pulso)
STATUS	OUTPUT	BOOL (I,Q,M,D,L)	Mostra estado se ERROR = 1
RD_1 ... RD_4	IN_OUT	ANY (I,Q,M,D)	Ponteiro da área de dados na CPU a qual os dados recebidos são armazenados. (SD_i e RD_i devem ser de acordo com os respectivos número, comprimento e tipo de dado.)

## SFBs de Comunicação: Bloco BSEND (SFB 12)

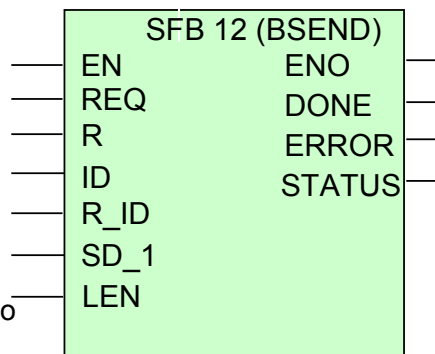
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL BSEND, I_BSEND //Com DB Instance
REQ:= I 0.4 //Partida
R:= I 0.5 //Reseta BSEND
ID:=W#16#3 //Conexão S7
R_ID:=DW#16#B2 //Bloco par
DONE:= #DONE_F //Término c/ sucesso
ERROR:= #ERROR_F //Término com erro
STATUS:= #STATUS_W //Informação adicional
SD_1:=P#DB1.DBX0.0 BYTE 40000 //Dado enviado
LEN:= #DB_LEN //Comprimento do dado
```

### Representação LAD

DB 12 (DB Instance)



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.27



Conhecimento em Automação  
Training Center

### Vista Geral

O SFB12 (BSEND) envia dados para um SFB parceiro remoto di tipo "BRCV". (O parâmetro R\_ID deve ser idêntico nos SFBs correspondentes.). Com este bloco pode-se transferir até 64 KByte de dados (aplicáveis a todas as CPUs).

A tarefa de envio é ativada após a chamada do bloco e quando existe transição positiva na entrada de controle REQ. A transmissão dos dados da memória do usuário é assíncrona ao processamento do programa do usuário.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL (I,Q,M,D,L constant)	Ativa uma transferência com uma transição positiva.
R	INPUT	BOOL (I,Q,M,D,L constant)	Ativa resetando BSEND para o estado inicial com uma transição positiva
ID	INPUT	WORD (I,Q,M,D,L constant)	Número da conexão para a conexão do sistema S7 individual (ver tabela de conexões)
R_ID	INPUT	DWORD (I,Q,M,D,L)	O parâmetro deve ser idêntico para ambos CFBs (BSEND e BRCV). Atribuição do bloco par
SD_1	IN_OUT	ANY (I,Q,M,D,L)	Dado a ser enviado, o comprimento no ponteiro any não é avaliado
LEN	IN_OUT	WORD (I,Q,M,D,L)	Comprimento do bloco de dados a ser transferido.
DONE	OUTPUT	BOOL (I,Q,M,D,L)	Sinaliza o erro –término livre do BSEND requisitado (pulso) com uma transição positiva.
ERROR	OUTPUT	BOOL (I,Q,M,D,L)	Transição positiva sinaliza um erro (pulso)
STATUS	OUTPUT	WORD (I,Q,M,D,L)	Contem instruções detalhadas de erros ou avisos.



## SFBs de Comunicação : Bloco BRCV (SFB 13)

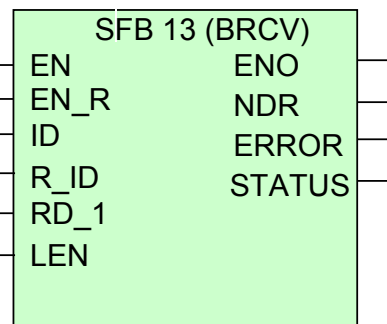
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL BRCV, I_BRCV           //Com DB Instance
EN_R:= I 0.4                //Partida
ID:=W#16#3                  //Conexão S7
R_ID:=DW#16#B2              //Bloco par
NDR:= #NDR_F                //Novo dado recebido
ERROR:= #ERROR_F            //Término com erro
STATUS:= #STATUS_W           //Informação adicional
RD_1:=P#DB2.DBX0.0 BYTE 40000 //Caixa correio Rec.
LEN:= #DB_LEN                //Comprim. cx.correio Rec.
```

### Representação LAD

DB 13 (DB Instance)



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.28



Conhecimento em Automação  
Training Center

### Vista Geral

O SFB13 (BRCV) recebe dados de um SFB parceiro remoto do tipo "BSEND". (O parâmetro R\_ID deve ser idêntico em ambos SFBs.) Após ele ter sido chamado e o valor 1 é aplicado na entrada de controle EN\_R, o bloco está pronto para receber dados. O endereço de partida da área de recepção é especificado pelo RD\_1.

Após o recebimento de cada segmento de dados, um reconhecimento é enviado ao SFB parceiro e o parâmetro LEN é atualizado. Se o bloco é chamado durante a recepção assíncrona dos dados, este fornece um aviso no parâmetro de saída STATUS; Se a chamada é feita quando o valor 0 é aplicado a entrada de controle EN\_R, a recepção está terminada e o SFB retorna seu valor de estado inicial. A recepção de segmentos de dados livre de erros é indicada pelo estado do parâmetro NDR tendo o valor 1.

Parâmetro	Modo	Tipo	Significado
EN_R	INPUT	BOOL (I,Q,M,D,Lconst.)	RLO = 1 SFB está pronto para receber. RLO = 0 procedimento está cancelado
ID	INPUT	WORD (I,Q,M,D,Lconst.)	Número da conexão da conexão do sistema S7 individual (ver tabela de conexão)
R_ID	INPUT	DWORD (I,Q,M,D,Lconst.)	Os parâmetros devem ser idênticos para ambos CFBs (BSEND e BRCV) . Atribuição do bloco par
RD_1	IN_OUT	ANY	Ponteiro para caixa de correio de entrada. O comprimento especificado determina o máximo comprimento dos blocos a serem recebidos. (para 2048 words)
LEN	IN_OUT	WORD	Comprimento dos dados recebidos até agora em bytes
NDR	OUTPUT	BOOL	Uma transição positiva sinaliza o programa do usuário: novos dados recebidos aceitos
ERROR	OUTPUT	BOOL	Uma transição positiva sinaliza erros (pulso)
STATUS	OUTPUT	WORD	Contem instruções detalhadas dos erro ou um aviso



## SFBs de Comunicação : Bloco STOP (SFB20)

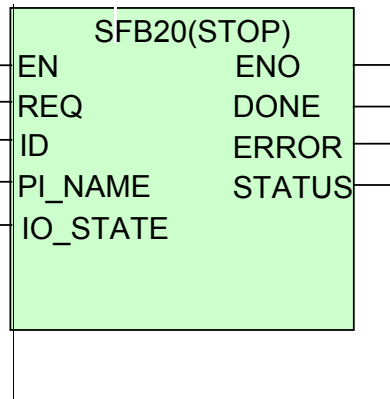
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL "STOP","I_STOP"      //DB Instance
REQ:= I 0.0               //Degrau de partida
ID:= W#16#1               //Número da conexão
PI_NAME:= P#M100.0 Byte 9 //Ver nota de rodapé
IO_STATE:=                //Não usado
DONE:= #DONE_F_20         //Término c/sucesso
ERROR:= #ERROR_F_20       //Término com erro
STATUS:= #STATUS_W_20     //Informações de erro
```

### Representação LAD/FBD

DB20 (DB Instance)



\* Detalhe da posição na memória para o início do: 'P\_PROGRAM'

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.29



Conhecimento em Automação  
Training Center

### Vista Geral

Se existe uma transição positiva na entrada de controle REQ, o SFB20 (STOP) ativa a mudança do estado STOP no equipamento remoto endereçado pelo ID. A mudança do modo é possível quando o equipamento está em RUN, HOLD ou modo startup.

A execução com sucesso da tarefa é indicada com estado 1 no parâmetro DONE. Se qualquer erro ocorrer eles são indicados nos parâmetros ERROR e STATUS.

A mudança para o novo modo somente pode ser iniciada novamente no mesmo equipamento remoto quando a primeira chamada do SFB 20 tiver sido completada.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL	Com uma transição positiva, ativa-se o STOP no equipamento endereçado pelo ID.
ID	INPUT	WORD (I,Q;M,D,L, constant)	Referencia a tabela e conexão ao número da conexão.
PI_NAME	IN_OUT	ANY	Ponteiro para área de memória na qual o nome do programa é iniciado (código ASCII) está locado. O nome deve ser P_PROGRAM para S7.
IO_STATE	IN_OUT	BYTE	Argumento de execução (não relevante)
DONE	OUTPUT	BOOL	Transição positiva = função executada
ERROR	OUTPUT	BOOL	Transição positiva = erro
STATUS	OUTPUT	WORD	Contem instruções detalhadas dos erros ou avisos (decimal)

## SFBs de Comunicação : Bloco START (SFB19)

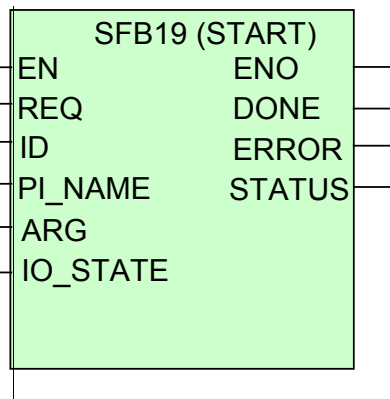
### Representação STL

Exemplo com parâmetros atribuídos

```
CALL "START","I_START" //Com DB Instance
REQ:= I 0.1           //Degrau de partida
ID:= W#16#1           //Número da conexão
PI_NAME:= P#M100.0 Byte 9 //Ver nota de rodapé
ARG:=                 //Não usado
IO_STATE:=            // Não usado
DONE:= #DONE_F_20     //Término com sucesso
ERROR:= #ERROR_F_20   //Término com erro
STATUS:= #STATUS_W_20 //Informações de erro
```

### Representação LAD/FBD

DB19 (DB Instance)



\* Detalhe da posição na memória para o início do: 'P\_PROGRAM'

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.30



Conhecimento em Automação  
Training Center

### Vista Geral

Se existe uma transição positiva na entrada de controle REQ, o SFB19 (START) ativa um restart completo no equipamento endereçado remoto pelo ID. As seguintes condições devem ser encontradas se o equipamento remoto é uma CPU:

- A CPU deve estar em estado STOP.
- A chave seletora da CPU deve estar em "RUN" ou "RUN-P".

Uma vez que o restart completo está finalizado, o equipamento muda para o modo RUN e envia a execução de um reconhecimento positivo. quando o reconhecimento positivo está disponível, o parâmetro de estado #DONE está setado em 1. Se qualquer erro ocorrer, eles são indicados pelo estado dos parâmetros #ERROR e #STATUS.

Parâmetro	Modo	Tipo	Significado
REQ	INPUT	BOOL	Ativa um restart completo no equipamento do ID endereçado com uma transição positiva.
ID	INPUT	WORD (I,Q;M,D,L, constant)	Referencia a tabela de conexão para o número da conexão.
PI_NAME	IN_OUT	ANY	Ponteiro para a área de memória na qual o nome do programa está iniciado (ASCII code) está presente. O nome P_PROGRAM deve estar presente para S7.
ARG	IN_OUT	ANY	Argumento de execução (não relevante)
IO_STATE	IN_OUT	ANY	Argumento de execução (não relevante)
DONE	OUTPUT	BOOL	Transição positiva = função executada
ERROR	OUTPUT	BOOL	Transição positiva = erro
STATUS	OUTPUT	WORD	Contem instruções detalhadas dos erros ou avisos (decimal)

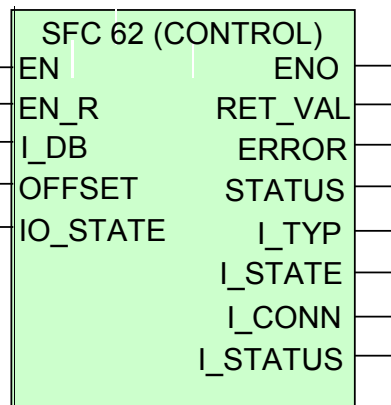
## SFBs de Comunicação : Bloco controle (SFC 62)

### Representação STL

Exemplo com parâmetros atribuídos

```
CALL "controle"
EN_R:= I 0. 2           //Partida
I_DB:= W#16#F           //Número DB Instance
OFFSET:= W#16#0         //Para multi-instances
RET_VAL:= MW4           //Informações erro
ERROR:= Q 0.4           //Término com erro
STATUS:= MW 4           //Informação estado
I_TYP:= MB 52           //Tipo de SFB
I_STATE:= MB 53         //Estado SFB
I_CONN:= M 54.0         //Estado de conexão
I_STATUS:= MW102        //Estado do SFB
```

### Representação LAD



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.31



Conhecimento em Automação  
Training Center

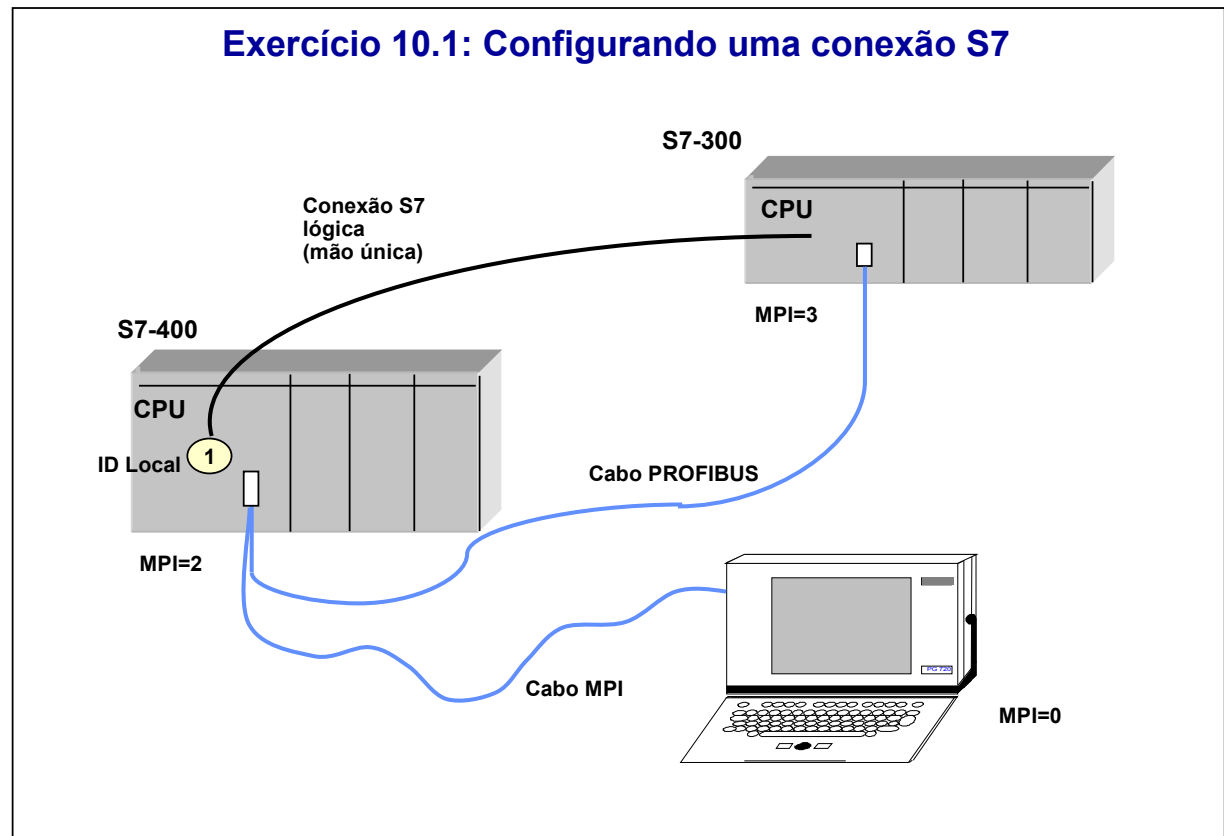
### Vista Geral

Com o SFC62 "CONTROL", você pode tomar o estado da conexão pertencente ao SFB instance a comunicação local.

Após a chamada da função do sistema com o valor 1 na entrada de controle EN\_R, o estado atual da conexão pertencente ao SFB instance de comunicação selecionado com I\_DB é suprido.

Parâmetro	Modo	Tipo	significado
EN_R	INPUT	BOOL	Parâmetro de controle para habilitação da função
I_DB	INPUT	BLOCK_DB (I,Q;M,D,L, constant)	Número do DB instance
OFFSET	INPUT	WORD (I,Q;M,D,L, constant)	Offset para multi-instances, número 1 <sup>st</sup> byte do DB instance (se nenhum multi-instance = 0)
RET_VAL	OUTPUT	INT (I,Q;M,D,L)	Erro 8000H para o SFC62
ERROR	OUTPUT	BOOL (I,Q;M,D,L)	RLO = 1 erro durante execução do SFC 62
STATUS	OUTPUT	WORD (I,Q;M,D,L)	Mostra erro para SFC 62
I_TYP	OUTPUT	BYTE (I,Q;M,D,L)	Identificador do tipo de CFB
I_STATE	OUTPUT	BYTE (I,Q;M,D,L)	Identificador do estado graph corrente do CFB
I_CONN	OUTPUT	BOOL (I,Q;M,D,L)	Estado da conexão atual 0 = conexão cancelada 1 = conexão presente
I_STATUS	OUTPUT	WORD (I,Q;M,D,L)	Erro ou estado do SFBs

## Exercício 10.1: Configurando uma conexão S7



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.32Conhecimento em Automação  
Training Center

### Tarefa

Rede de comunicação entre estações S7-400 e S7-300 e configuradas como uma conexão S7.

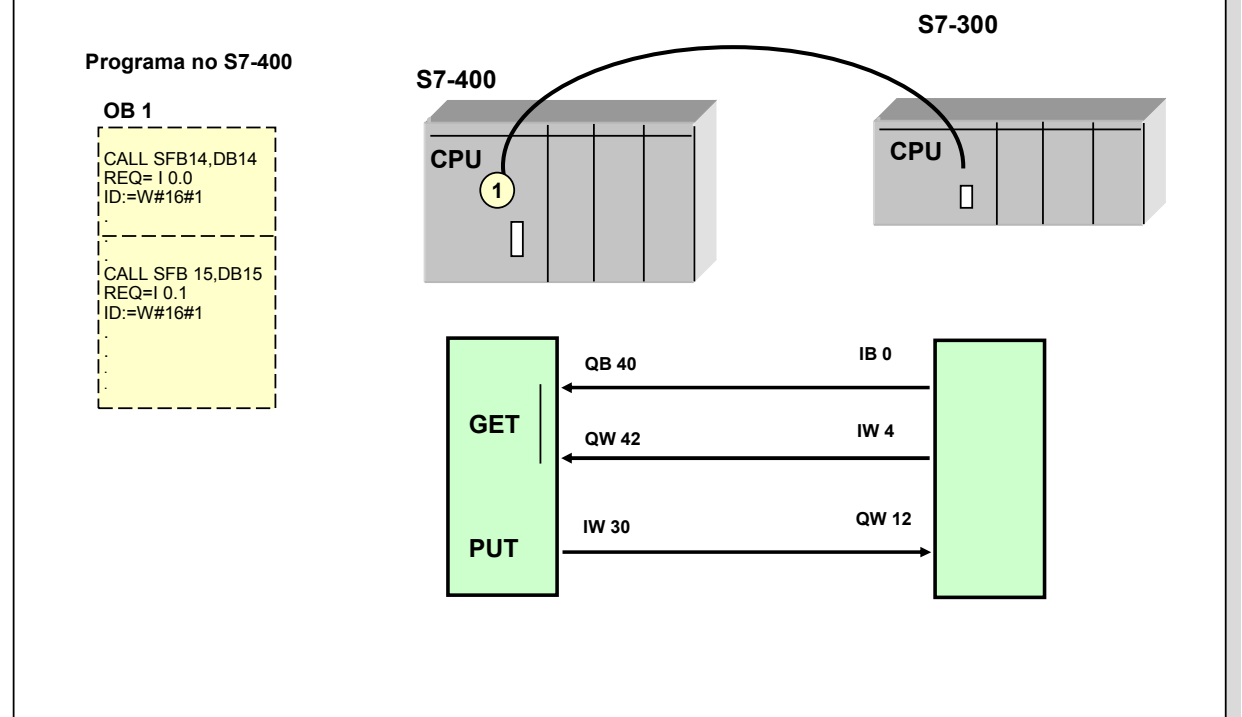
### O que fazer

1. Criar um novo projeto "SFB-Comm".
2. Gerar duas estações de HW para S7-400 e S7-300 no seu projeto.
3. No *HW Config* configurar diferentes endereços MPI para as duas CPUs e "Network" (colocar em rede) as duas CPUs com o objeto "MPI network" em seu projeto.
3. Então transfira os dados de configuração individualmente para cada CPU utilizando a ferramenta *HW Config*.
4. Colocar em rede as duas estações através do cabo MPI e verifique o resultado utilizando a função da PG: "Accessible Nodes".
5. Configurar uma conexão S7 entre as duas CPUs e transfira a tabela de conexão compilada para a CPU S7-400.
6. Utilizando a opção de menu PLC -> Modul Information, verificar qual é a conexão que foi atualmente reservada na CPU S7-400 (Register: *Communication -> Reserved Connection*)
7. Executar um restart completo no S7-400.
8. Verificar qual é a conexão reservada que foi estabelecida. Para isto, ler a informação do estado Online da CPU S7-400 utilizando a opção de menu PLC -> Modul Information.  
Então verificar no registrador de comunicação qual é a conexão reservada que foi estabelecida.

### Nota

O S7-300 não tem qualquer dado de configuração e dados online que irão fornecer a informação sobre as conexões reservadas e atuais.

## Exercício 10.2: Comunicação com os SFBs GET/PUT



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.33Conhecimento em Automação  
Training Center

### Tarefa

Para o S7-400, criar um OB1 com a seguinte funcionalidade:

- Através da entrada 28.0, IB0 e IW 4 do S7-300 pode ser lido e transferido para as saídas QB40 ou QW42 do S7-400 local.
- Através da entrada 28.1, a IW30 do S7-400 pode ser escrito no QW12 no S7-300.

### O que fazer

1. Gerar uma pasta S7 Program com o nome: SFB\_GET\_PUT.
2. Editar o OB1. Gerar uma rede de comunicação "SFB\_GET", na qual você chame o SFB "GET" (gatilho I28.0).  
Na chamada do "GET" ler o conteúdo do IB0 do S7-300 e colocar o valor na saída QB40 do S7-400.  
Também ler o conteúdo da IW4 e colocar na QW42 do S7-400.
3. Gerar uma rede de comunicação "SFB\_PUT" e chamar o SFB "PUT" (gatilho I28.1).  
Na chamada do "PUT" transferir a IW2 do S7-400 para QW12 do S7-300.
4. Transferir o parâmetro de saída STATUS (pulso) dos SFBs para o display digital (QW38) do S7-400.
5. Transferir o OB1 da CPU S7-400 e testar seu programa.

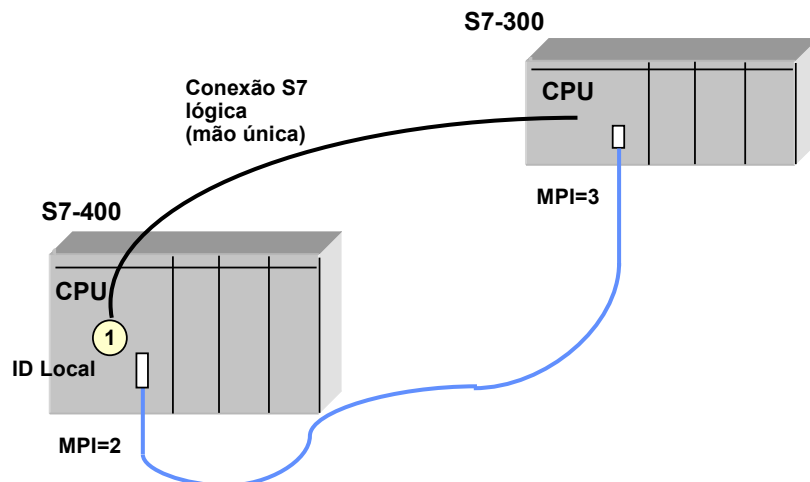
## Exercício 10.3: Comunicação com os SFBs START/STOP

### Programa no S7-400

#### OB 1

```
CALL SFB20,DB20
REQ= I 28.0
ID:=W#16#1
PI_NAME:= P#M100.0 Byte 9

CALL SFB 19,DB19
REQ= I 28.1
ID:=W#16#1
PI_NAME:= P#M100.0 Byte 9
```



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_10P.34



Conhecimento em Automação  
Training Center

### Tarefa

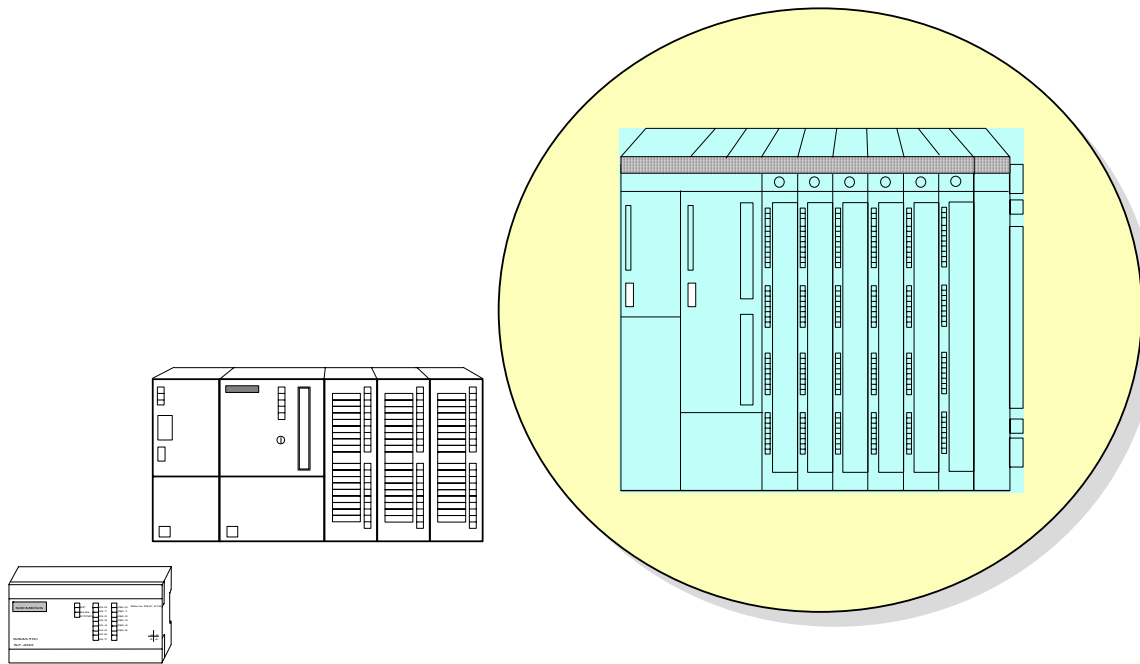
Para o S7-400, criar um OB1 com as seguintes funcionalidades:

- O parceiro meta (S7-300) pode ser "stopped" através da entrada 28.0
- O parceiro meta pode ser "started" através da entrada 28.1.

### O que fazer

1. Gerar uma pasta S7 Program com o nome: "SFB\_START\_STOP"
2. Editar o OB1. Gerar uma rede de comunicação "P\_PROGRAM", na qual você guarde os caracteres (individuais) "P\_PROGRAM" da MB100 para MB109.
3. Gerar uma rede de comunicação "SFB\_STOP", na qual você chame o SFB "STOP" (gatilho 28.2).
4. Gerar uma rede de comunicação "SFB\_START", na qual você chame o SFB "START" (gatilho 28.3).
5. Transferir o parâmetro de saída STATUS (pulso) dos SFBs para o display digital (QW38) do S7-400
6. Transferir o OB1 para a CPU S7-400 e teste o seu programa.

## SIMATIC S7-400



## SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.1Conhecimento em Automação  
Training Center

## Conteúdo

## Pág.

SIMATIC S7-400: Vista Geral .....	2
Vista geral do Módulo S7-400 .....	3
Os bastidores do S7-400 .....	4
Multiprocessamento simétrico e assimétrico .....	5
Configuração Centralizada .....	6
Parâmetros dos Módulos: endereços lógicos, tabelas de Imagens de Processo .....	7
Atribuição de Parâmetros dos Módulos: Módulo Analógico .....	8
Configurando o Multiprocessamento .....	9
SFC 35 para sincronização em Modo Multiprocessamento.....	10
Expansão Centralizada 1 .....	11
Expansão Centralizada 2 .....	12
Expansão Distribuída .....	13
Conexão Distribuída entre S7 e S5 .....	14
Expansão de uma Configuração Centralizada .....	15
Módulos CPU .....	16
Dados Técnicos das CPUs S7-400 (1) .....	17
Dados Técnicos das CPUs S7-400(2) .....	18
Arquitetura do Sistema .....	19
Parâmetros da CPU: Características de Startup .....	20
Parâmetros da CPU: Interrupções .....	21
Parâmetros da CPU: Dados Locais .....	22
Parâmetros da CPU: Conceitos de Proteção .....	23
Organização de Programa : Restart Completo e Restart .....	24
A Interrupção de Inserção/Remoção de Módulos no S7-400 .....	25
O Comando Force no S7-400 .....	26
Ativando a Barra Breakpoint .....	27
Execução de Programas com Breakpoints (somente S7-400) .....	28
Habilitação de Saídas de Periferia (somente S7-400) .....	29
CP 441 para Conexões Ponto-a-Ponto .....	30
CP 443-5: Conexão para PROFIBUS .....	31
IM 467: Interface Mestre PROFIBUS-DP .....	32
CP 443-1: Conexão para Ethernet Industrial .....	33
CP 443-1 IT: Conexão para Internet .....	34

## SIMATIC S7-400: Vista Geral

### Expansibilidade do Sistema

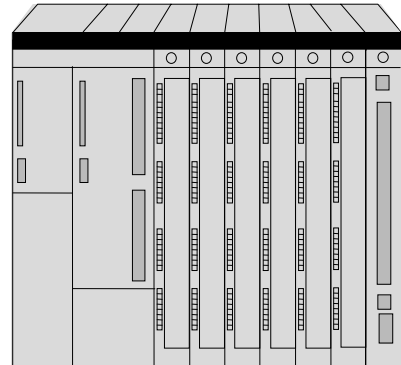
- Alta Densidade de Encapsulamento
- CPUs com Performance Graduada
- Multiprocessamento
- Podem ser conectados até 21 bastidores de expansão
- Extensa gama de módulos (SMs, FMs, CPs)
- Flexibilidade e facilidade de construção de redes de comunicação

### Performance

- Alto Poder de Processamento (até 80 nseg por instrução binária)
- Até 20 Mbytes de memória do usuário
- Facilidades de comunicação poderosas

### Versatilidade

- Funções especiais
- Especial facilidade de migração para S5



## SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.2



Conhecimento em Automação  
Training Center

### SIMATIC S7-400

O SIMATIC S7-400 é o PLC para média até elevada faixa de performance. O projeto modular e sem ventilação, a alta capacidade de expansão e robustez, as extensas possibilidades de comunicações e a alta performance o tornam disponível para qualquer necessidade de projeto.

### Expansibilidade

Os pontos especiais do S7-400 são:

- Módulo de montagem simples em técnica "swing-out". Todos os módulos são para operação sem ventilação e oferecem uma alta densidade de encapsulamento. Comparado com o S5, o espaço de montagem é reduzido em 54%, o espaço por conexão de I/O é reduzido em 45%.
- O S7-400 oferece uma performance escalonável através de um espectro graduado de módulos de CPU disponíveis, bem como através de capacidade de multiprocessamento simétrica e assimétrica.
- Uma extensa variedade de módulos, isto é, para cada aplicação uma CPU disponível, módulos de sinal, módulos de função e módulos de comunicações.
- Arquitetura aberta com até 21 bastidores de expansão e unidades adicionais distribuídas através de PROFIBUS DP.
- Através das possibilidades de montagem de redes através de MPI, PROFIBUS e Ethernet Industrial, o S7-400 também está disponível para tarefas de controle de processo.

### Performance

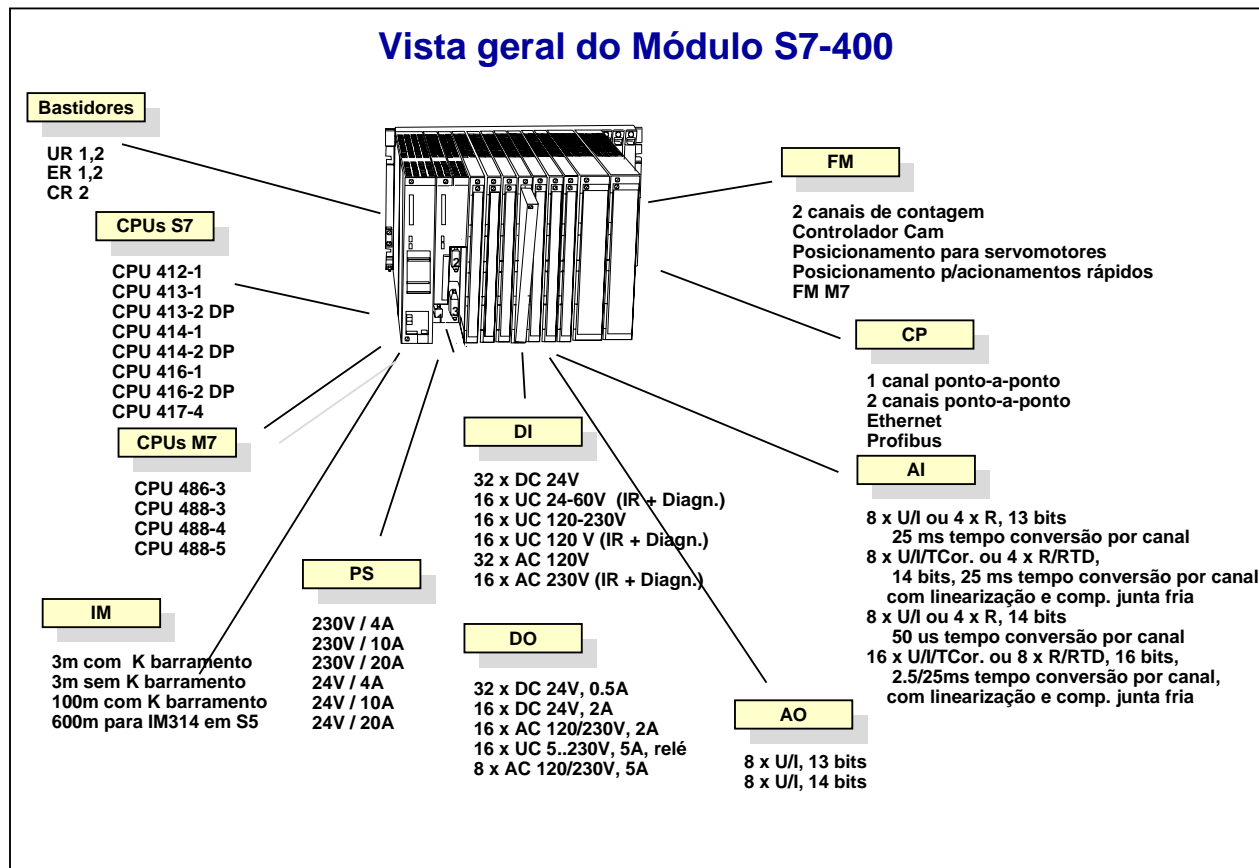
- A alta velocidade de processamento com o mínimo tempo de 80 ns por instrução e uma memória do usuário de até 1.6 MB torna possível a realização de extensas tarefas de automação.
- A alta performance do barramento de comunicação (10.5 Mbaud) garante comunicação rápida com alta troca de dados.

### Versatilidade

- Versatilidade através de funções especiais, como por exemplo: restart, remoção e inserção de módulos em RUN, etc.
- Adicionalmente o S7-400 oferece possibilidades especiais de migração do S5 para S7, como:
  - utilização de IPs ou WFs do S5 nos bastidores centrais S7;
  - conexão de unidades de expansão S5 para um bastidor central S7;



## Vista geral do Módulo S7-400



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.3



Conhecimento em Automação  
Training Center

#### Bastidores

Os seguintes bastidores estão disponíveis para o S7-400.

- UR1/UR2 são projetados como bastidores universais e podem ser utilizados como um bastidor central ou como um bastidor de expansão. São construídos com 18/9 slots largura simples com os barramentos P e K;
- ER1/ER2 são bastidores de expansão sem barramento K;
- CR2 é um bastidor central segmentado para multiprocessamento simétrico;

#### CPUs S7 :

As CPUs S7-400 são compatíveis para cima com todos os programas do usuário em STEP 7. Elas estão disponíveis em duas versões: largura simples e largura dupla com interface DP Mestre integrada.

Um máximo de 64 estações DP escravas podem ser endereçadas através da interface DP integrada. A máxima velocidade de comunicação é de 12 Mbaud.

#### FMs

Os FMs para posicionamento, controle em malha fechada e contagem substituem o espectro dos IPs do S5.

Adicionalmente, um FM M7 pode ser inserido como um módulo de função programável livremente em C para controle de processo.

#### IMs

Os bastidores de expansão SIMATIC S7 e SIMATIC S5 podem ser conectados a um bastidor central S7-400 através dos módulos de interface.

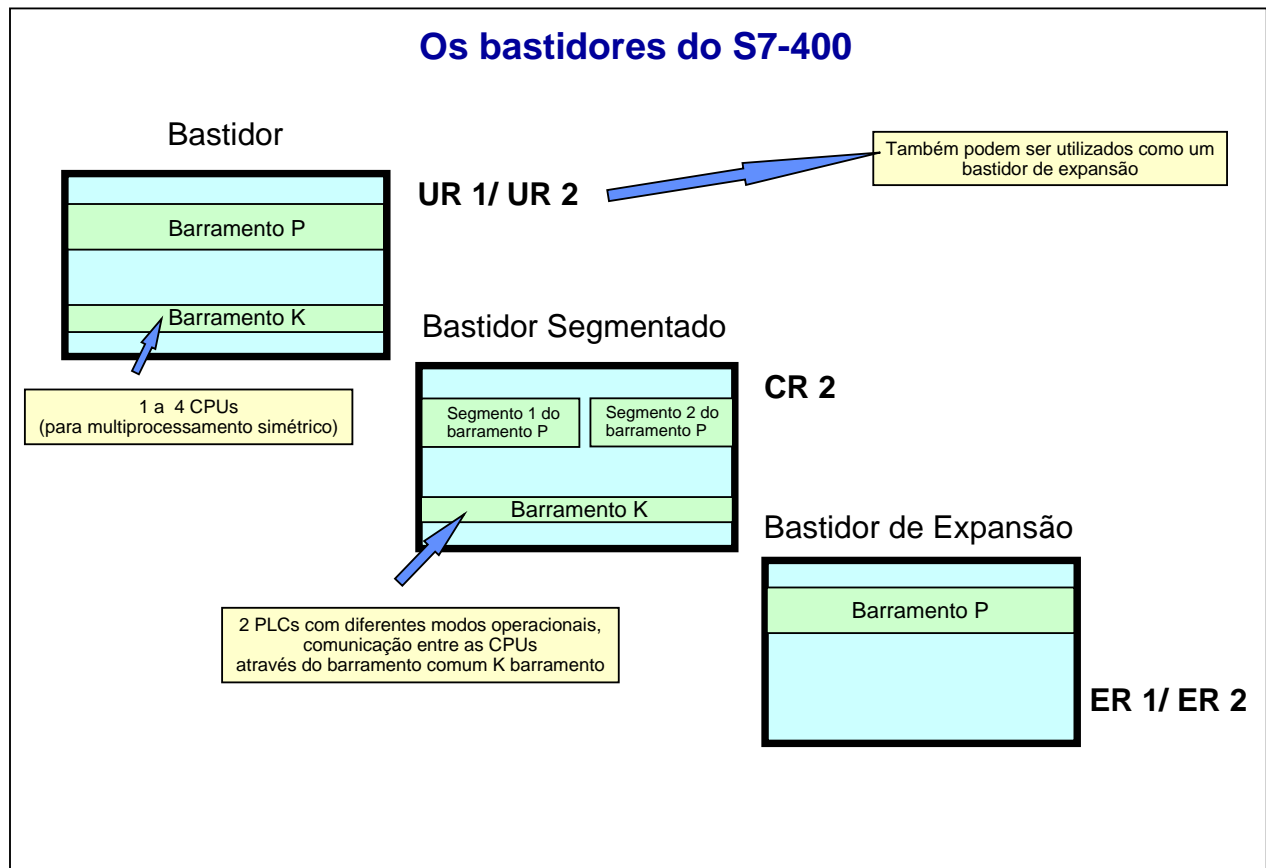
#### CPs

Os módulos CP fazem a conexão de uma CPU às seguintes possíveis redes de comunicação:

- Ethernet Industrial (CP 443-1);
- PROFIBUS (CP 443-5);
- Rede de Comunicação Ponto-a-Ponto (CP441-1 e CP441-2);

Mais ainda, cada CPU disponibiliza de uma interface MPI para conexão à uma rede MPI. Um máximo de 32 estações podem ser conectadas a uma rede MPI.

## Os bastidores do S7-400



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.4



Conhecimento em Automação  
Training Center

#### UR 1 / UR 2

Ambos os UR1/UR2 podem ser utilizados como bastidor central e como expansão. Eles tem um barramento paralelo de I/O (P barramento) para a alta velocidade de troca de sinais de I/O (1.5 microseg./ byte) e tempo crítico de acesso aos dados de processo pelos módulos de sinal.

Adicionalmente, UR1 (18 slots) e UR2 (9 slots) possuem um poderoso barramento serial de comunicação (K barramento) para troca de dados de alta velocidade (10.5 Mbaud) entre estações K barramento (S7/M7 CPUs, FMs, CPs).

Pela separação dos P barramento e K barramento, cada tarefa é atribuída a seu próprio sistema de barramento. Controle e comunicação tem seu próprio barramento de dados de alta velocidade individual, fornecendo então filtragem e controle livre de conflitos para operações de comunicação.

#### CR2

O bastidor segmentado CR 2 fornece um barramento de I/O dividido em dois segmentos com 10 e 8 slots. Uma CPU pode ser utilizada em cada segmento. Cada CPU é Mestre em seu respectivo segmento P barramento e pode somente acessar seus próprios SMs.

A transição do modo de operação não é sincronizada, isto é, as CPUs podem estar em diferentes modos de operação. Ambas CPUs podem comunicar através do barramento contínuo K barramento.

#### ER 1 / ER 2

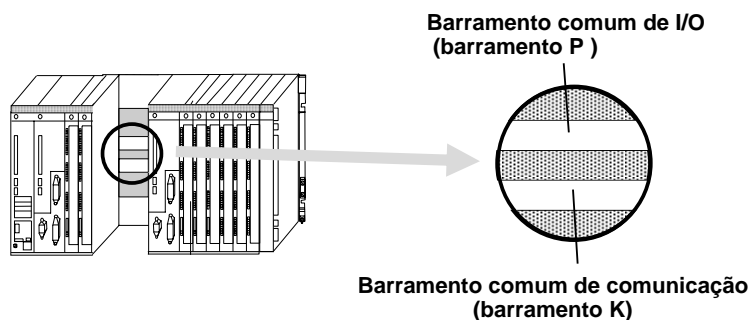
Os ER1 (18 slots) e ER2 (9 slots) não possuem K barramento, nenhuma linha de interrupção, nenhuma alimentação 24 V para os módulos e nenhuma alimentação de bateria.

#### Sem regras p/Slots

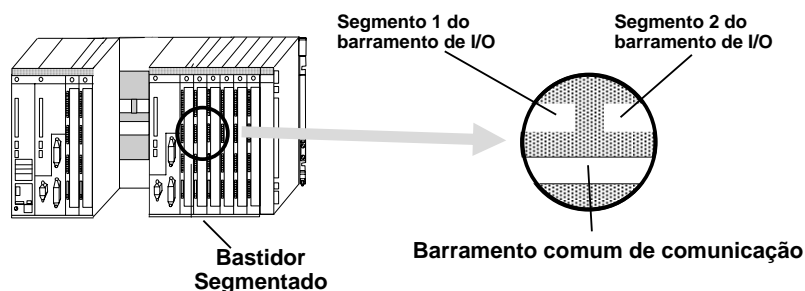
Exceção: PS no lado extremo esquerdo e IM de recepção no ER no lado extremo direito.

## Multiprocessamento simétrico e assimétrico

### • Multiprocessamento Simétrico



### • Multiprocessamento Assimétrico



## SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.5



Conhecimento em Automação  
Training Center

### Multiprocessamento

Multiprocessamento torna um sistema PLC escalonável, isto é, ele habilita a performance e os recursos do sistema, ou seja, memória, memórias de bits, temporizadores, contadores, etc. para serem incrementados. Desta forma, por exemplo, uma tarefa complexa pode ser dividida ao longo de diversas CPUs.

### Simétrica

No multiprocessamento simétrico, todas as CPUs (máx. 4 CPUs) dividem um barramento comum P barramento e K barramento. Em particular, existe somente um espaço de endereço de I/O comum no qual os endereços de todos os módulos de sinal são mapeados. Cada módulo inserido deve deste modo ser atribuído a uma CPU durante a configuração. A CPU então assume como "Função Mestre" para este módulo, como:

- Recepção de interrupções do módulo
- Atribuição dos parâmetros do módulo
- Acesso aos módulos através de L PBxx, T Wxx, etc.

As transições do modo de operação são sincronizadas, isto é, todas as CPUs tem o mesmo modo de operação. Sendo vista pelo lado de fora, a estação aparece como sendo um grande PLC.

### Assimétrica

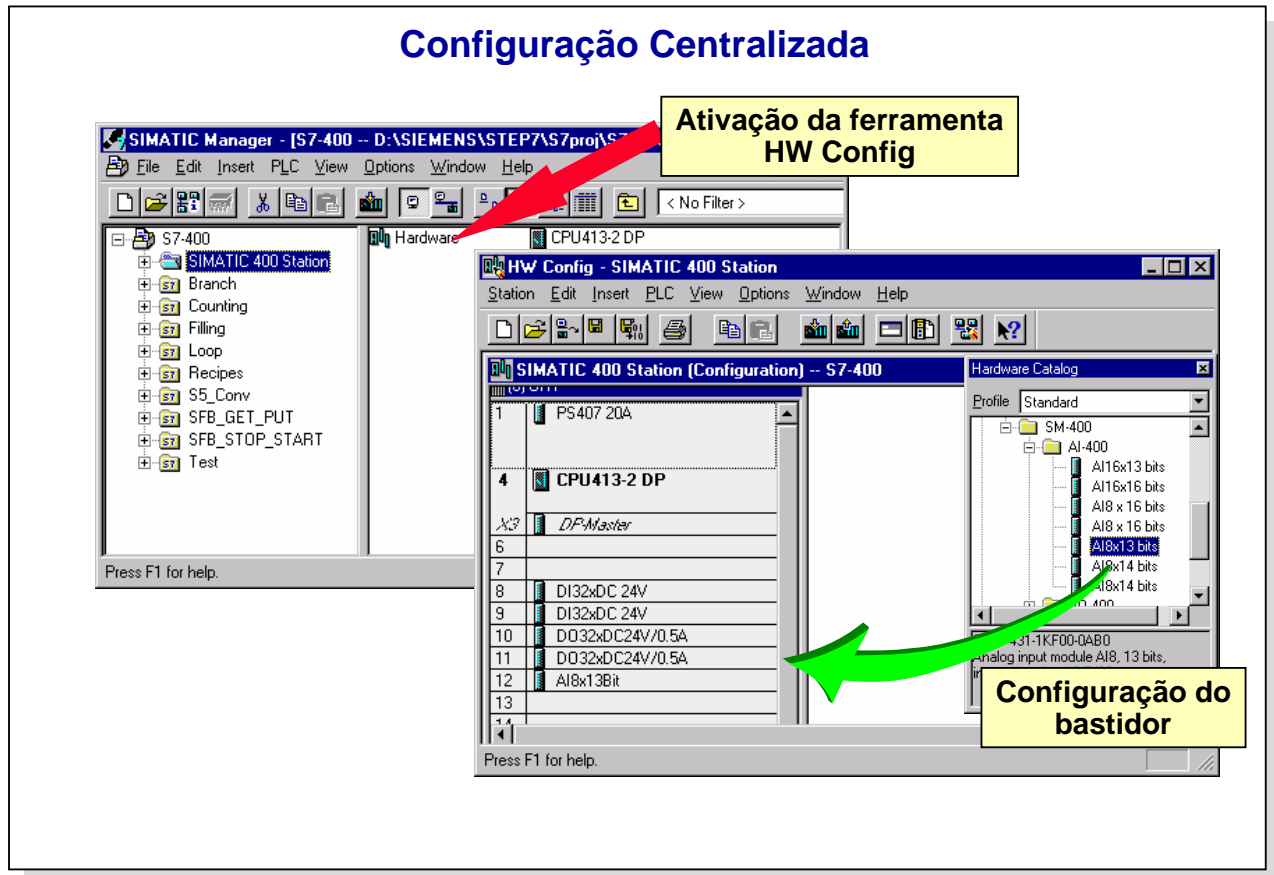
Multiprocessamento assimétrico é atingido com a ajuda do CR2. O bastidor segmentado contém dois segmentos de barramento P barramento independentes.

Uma CPU é instalada por segmento de barramento de I/O. Os módulos de I/O são localmente atribuídos a esta CPU. As CPUs trabalham independentemente uma da outra sem sincronização das transições dos modos de operação. Cada CPU tem seu próprio espaço de endereços de I/O.

O barramento de comunicação comum faz comunicação entre as duas unidades possíveis sem o uso de hardware adicional. Sendo vista pelo lado de fora, este corresponde a dois controladores individuais, que comunicam através do K bus. Demais vantagens são:

- Economia de espaço no painel de controle;
- Mais economia, uma vez que somente um bastidor e uma fonte de alimentação são necessárias;

## Configuração Centralizada



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.6



Conhecimento em Automação  
Training Center

### Configurando uma estação centralizada

Para uma configuração centralizada, disponha os módulos próximos a CPU(s) no bastidor central e continue a inserí-los nos bastidores de expansão seguintes.

### Criando uma configuração

Para abrir a configuração de uma estação proceda como abaixo:

1. Antes de mais nada, selecione com um clique no mouse a estação de hardware desejada.
2. Selecione a opção do menu *Edit -> Object*, abra ou dê um duplo clique no símbolo *Hardware* na janela do lado direito. A janela estação da estação selecionada é aberta.
3. Um clique no símbolo do Catálogo mostra o Catálogo de HW com os components disponíveis. Deste catálogo copie por "marcar e arrastar" o bastidor e os módulos dentro da janela estação ou dentro da tabela de configuração do respectivo bastidor.



### Vista do Catálogo

Um clique no sinal "+" abre a subestrutura associada ou com um clique no sinal "-" este é fechado. Após a seleção de um módulo, o dado técnico mais importante do módulo aparece na barra de status da janela catálogo.

### Selecionando o bastidor

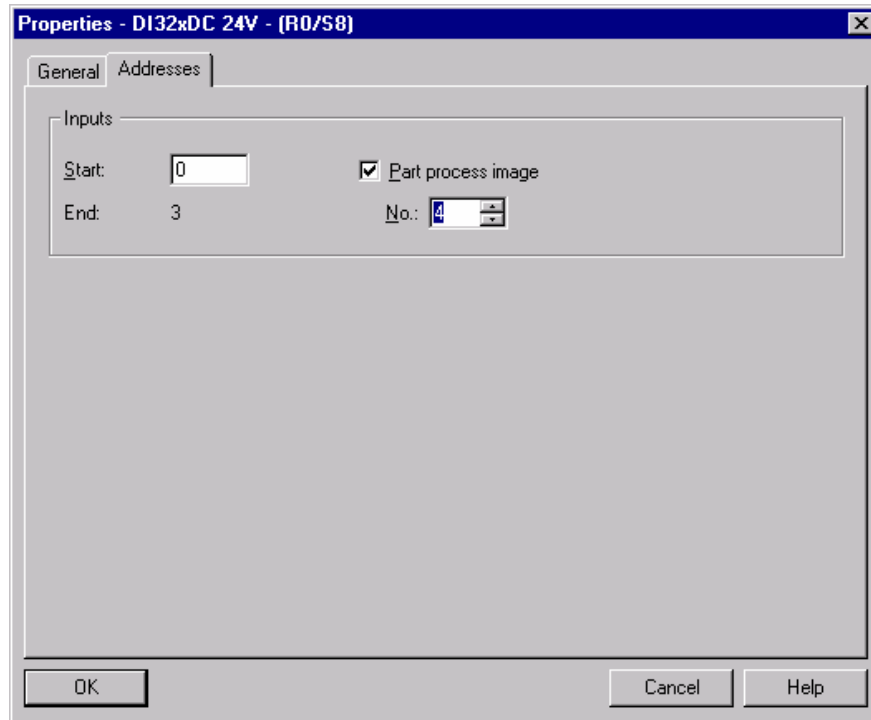
Quando você está com a janela da estação aberta e o catálogo de hardware, você pode continuar como abaixo:

1. Dependendo do tipo de estação, selecione o S7-400 inserido.
2. Antes de mais nada, abra o bastidor inserido e arraste o bastidor para dentro da estação na janela da esquerda. Uma tabela vazia é mostrada para cada bastidor.

Bastidores estão representados pelas tabelas de configuração no STEP 7. Estas tabelas de configuração possuem um número de linhas de entradas igual ao número de módulos que podem ser instalados no bastidor.

3. Então, usando "marcar e arrastar" copie os módulos desejados para a esquerda dentro dos slots vazios da tabela.

## Parâmetros dos Módulos: endereços lógicos, tabelas de Imagens de Processo



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.7



Conhecimento em Automação  
Training Center

#### Geral

No sistema S7-400 existem endereços padrão para os módulos de I/O. Estes padrões estão ativos até que os dados parametrizados sejam transferidos para a CPU.

#### Endereços padrão

Os endereços padrão dos módulos dependem do:

- Número do bastidor. O número é ajustado na IM de recepção com uma chave (1..21), o bastidor central sempre tem o número 0;
- Os slots dos módulo no bastidor. O endereço padrão de um módulo é calculado destes dois valores como segue:

Endereço Digital inicial = [(bastidor número) x 18 + número do slot - 1] x 4

End. Analóg. inicial = [(bastidor número) x 18 + número do slot - 1] x 64 + 512

#### Imagem de

#### Processo Parcial

Ao lado da imagem de processo completa, você pode atribuir parâmetros

para até 8 imagens processo parciais. As imagens de processo parciais podem ser atualizadas no programa do usuário através de funções do sistema (SFC 26/27).

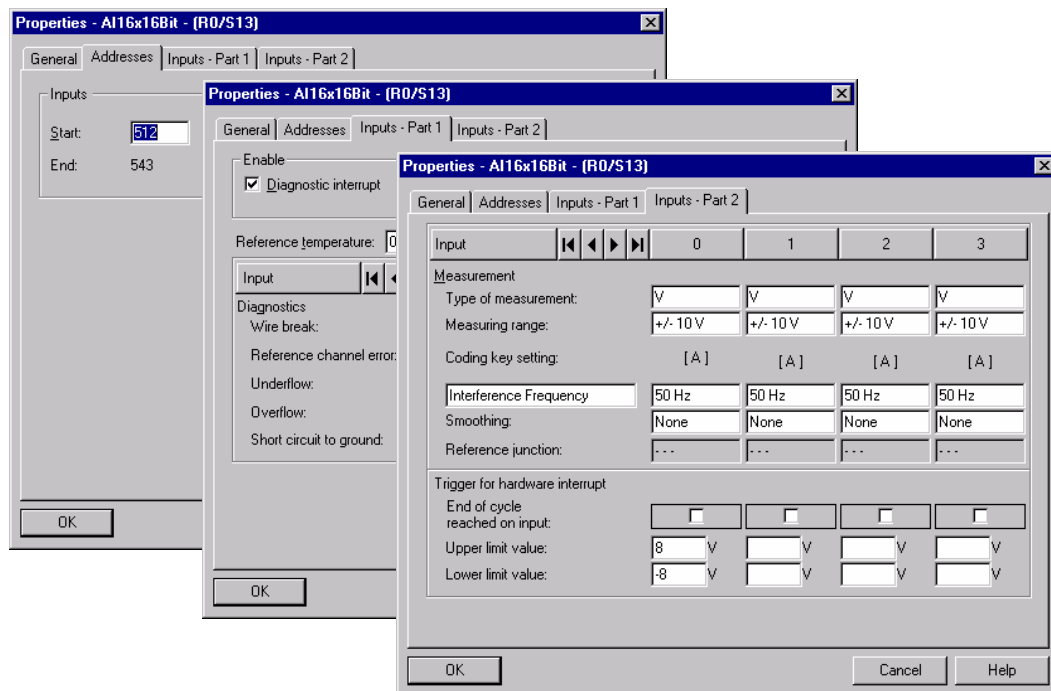
Deste modo, o usuário tem a possibilidade de usar o conceito de imagem de processo, que foi projetada originalmente somente para a classe de prioridade OB1, para outras classes de prioridade, por exemplo OB35 para algoritmos de controle.

Para isto, os módulos de entrada bem como os de saída os quais lidam com o valor atual ou com o valor do setpoint do algoritmo corrente de controle são em cada caso atribuídos a uma tabela de imagem de processo parcial de entrada ou saída.

Com o OB35 você pode então proceder da seguinte maneira:

1. Lendo o valor corrente atual por meio do SFC26 na tabela associada da imagem de processo de entrada parcial.
2. Chamando o algoritmo de controle. O algoritmo de controle escreve seu novo valor de setpoint na correspondente tabela de imagem de processo parcial de saída.
3. Por para fora tabela de imagem de processo parcial de saída por meio do SFC27 para as I/Os.

## Atribuição de Parâmetros dos Módulos: Módulo Analógico



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.8



Conhecimento em Automação  
Training Center

### Módulos c/atribuição de parâmetros

Todos os módulos que possam ter atribuição de parâmetros, por exemplo módulos analógicos, podem ter parâmetros atribuídos com a ferramenta HW Config.

Para módulos analógicos, existem usualmente diversas páginas de tabelas para atribuição de parâmetros. Desta forma, os seguintes parâmetros podem ser setados, por exemplo nas páginas da tabela individual de um módulo analógico S7-400.

### Endereços

- Endereço de partida do módulo
- Número da parte da imagem de processo
- OB de interrupção de hardware

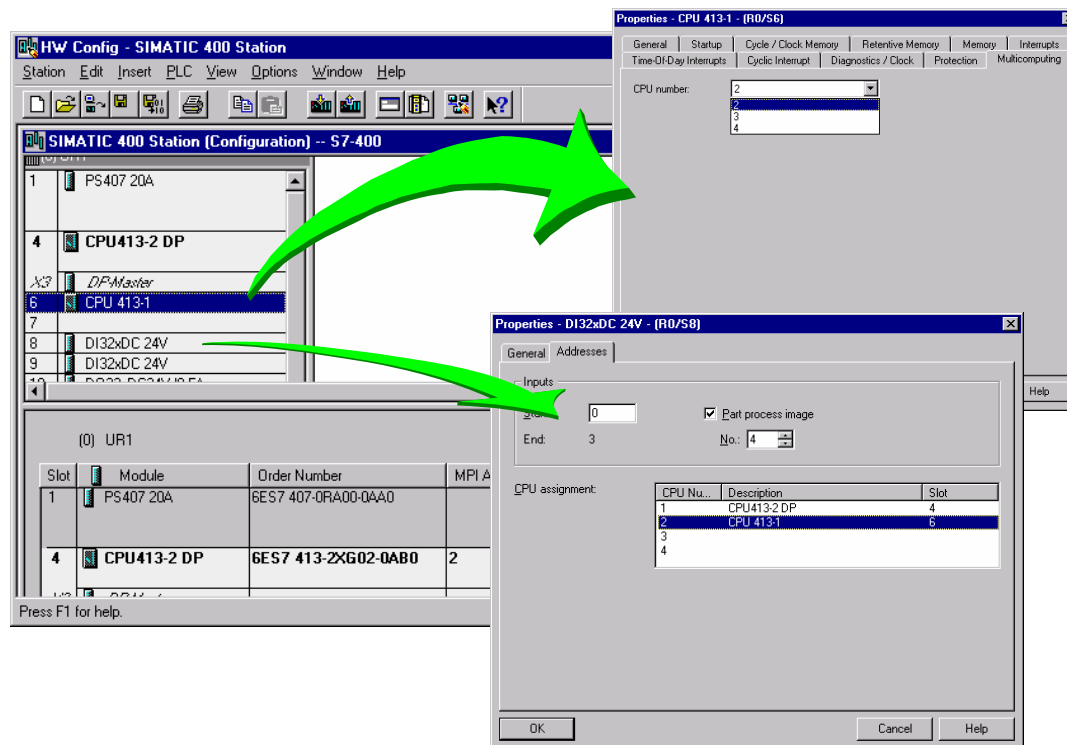
### Inputs - Parte 1

- Habilitação para interrupção de hardware e diagnóstico de interrupção
- Habilitação de diagnóstica da monitoração de canal específico, como
  - Verificação de quebra de fio
  - Erro no canal referenciado
  - Ultrapassagem de valor inferior (underflow)
  - Ultrapassagem de valor superior (overflow)
  - Curto circuito à terra

### Inputs - Parte 2

- Tipo de medição
- Faixa de medição
- Supressão de frequência de interferência
- Filtro passa baixas de entrada
- Valor limite superior e inferior para interrupção de hardware

## Configurando o Multiprocessamento



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.9



Conhecimento em Automação  
Training Center

### Vista Geral

Multiprocessamento é a operação síncrona de diversas CPUs (2 a 4) no bastidor central S7-400.

As CPUs inicializam em conjunto, se elas possuem o mesmo modo de startup (restart completo ou restart) e elas também vão para o modo STOP juntas.

### Ajustes para Multiprocessamento

Multiprocessamento aparece implicitamente através da inserção de diversas CPUs com capacidade de multiprocessamento em um bastidor apropriado. Se uma CPU é capaz de multiprocessamento pode ser determinado no texto informativo no "Hardware Catalog" (catálogo de hardware).

Uma área de endereço comum está dividida entre as CPUs participantes no multiprocessamento, isto é, a área de endereço de um módulo está sempre exclusivamente associado com uma CPU.

### Procedimento

Multiprocessamento é configurado como segue:

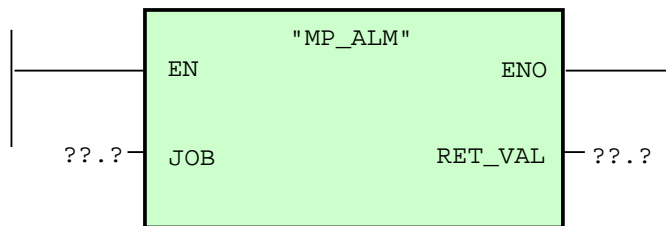
1. Posicionar todas as CPUs requeridas para multiprocessamento.
2. Dê um duplo clique nas CPUs e ajuste o número da CPU na página tabela de "multiprocessamento".
3. De forma a atribuir um módulo para uma CPU particular, proceda como abaixo:
  - Arrange os módulos no bastidor.
  - Dê um duplo clique nos módulos e selecione a tabela de "endereços".
  - No campo "CPU No." selecione o número da CPU desejada. Com os módulos gatilháveis por interrupção, a atribuição da CPU é mostrada como a CPU meta na página de "Inputs" ou "Outputs".

Através do comando *View -> Filter -> CPUn moduls*, você pode fazer os módulos que estão atribuídos para uma CPU particular aparecer em uma tabela.

A estação configurada somente pode ser transferida para todas as CPUs. Transferência para somente uma CPU não é possível. Deste modo, configurações inconsistentes são evitadas.



## SFC 35 para sincronização em Modo Multiprocessamento



Parâmetro	Declaração	Tipo dado	Área memória	Descrição
JOB	INPUT	BYTE	I, Q, M, D, L, Const.	Identificador de Job (valores possíveis : 1 a 15)
SFB 9	URCV	Mão dupla	Mão dupla	Mão dupla

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.10



Conhecimento em Automação  
Training Center

### Descrição

A chamada do SFC 35 "MP\_ALM" gatilha a interrupção de multiprocessamento no modo multiprocessamento. Isto comanda a partida sincronizada do OB60 em todas as CPUs associadas.

Na operação de processador simples e operação em bastidores segmentados, OB60 é somente disparado na CPU no qual o SFC 35 é chamado.

Com o parâmetro de entrada JOB, você pode identificar a causa para a interrupção multiprocessamento. Este job identificador é transmitido para todas as CPUs associadas e pode ser avaliado através da informação de partida no OB 60.

Você pode chamar o SFC 35 "MP\_ALM" de qualquer localização em seu programa. Desde que a chamada somente faz sentido no modo RUN, a interrupção de multiprocessamento é suprimida quando chamada no modo STARTUP. Isto é comunicado a você através do valor retornado.

### Código de Erro

Se um erro ocorre enquanto a função está sendo processada, o valor retornado contém um código de erro:

W#16#0000: Nenhum erro.

W#16#8090: O parâmetro de entrada JOB contém um valor não permitido.

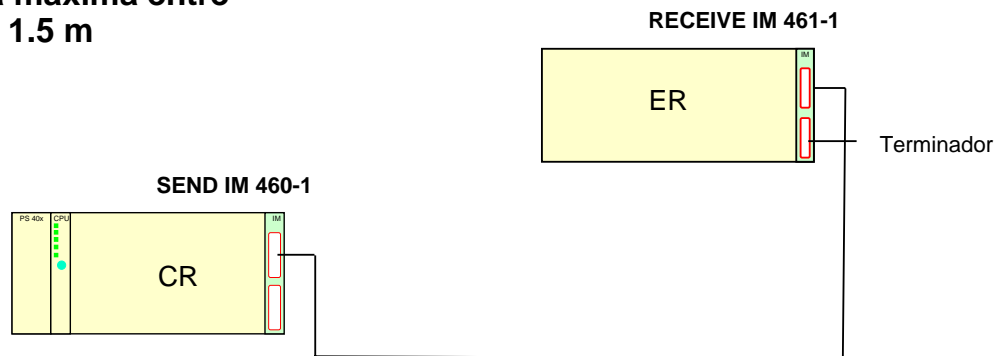
W#16#80A0: Processamento da interrupção prévia de multiprocessamento com o OB 60 não tenha ainda sido completada na CPU local ou em outra CPU.

W#16#80A1: Modo de operação incorreta (STARTUP em vez de RUN).



## Expansão Centralizada 1

- Barramento P e conectado através de fonte de alimentação, mas não o barramento K
- 1 ER por canal
- Distância máxima entre CR e ER: 1.5 m



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.11



Conhecimento em Automação  
Training Center

**Config. centralizada 1** Os módulos de interface IM 460-1/IM 461-1 são inseridos para a conexão centralizada dos bastidores de expansão para um bastidor central S7.

Expansão centralizada tipo 1 tem as seguintes características:

- Um máximo de bastidores de expansão 2 pode ser conectado (1 por interface).
- A máxima distância entre bastidor central e de expansão é de 1,5 m.
- Um máximo de 2 Send IM 460-1 pode ser inserido por bastidor central.
- O módulo de interface de envio IM 460-1 conecta somente o barramento P (não o barramento K) através do bastidor de expansão. Além disso, os módulos no bastidor de expansão são alimentadas com uma tensão de 5V (máx. 5 A por slot) através de cabo de conexão.

Por esta razão, nenhum módulo de fonte de alimentação pode ser inserido no bastidor de expansão.

- Um conector de interface não ocupado no Send IM 460-1 não tem que ser terminado; um conector de interface não ocupado no Receive IM 461-1 deve ser terminado com um terminador.
- Uma chave de código com a qual o número do bastidor de expansão deve ser selecionado está localizado no Receive IM 461-1.

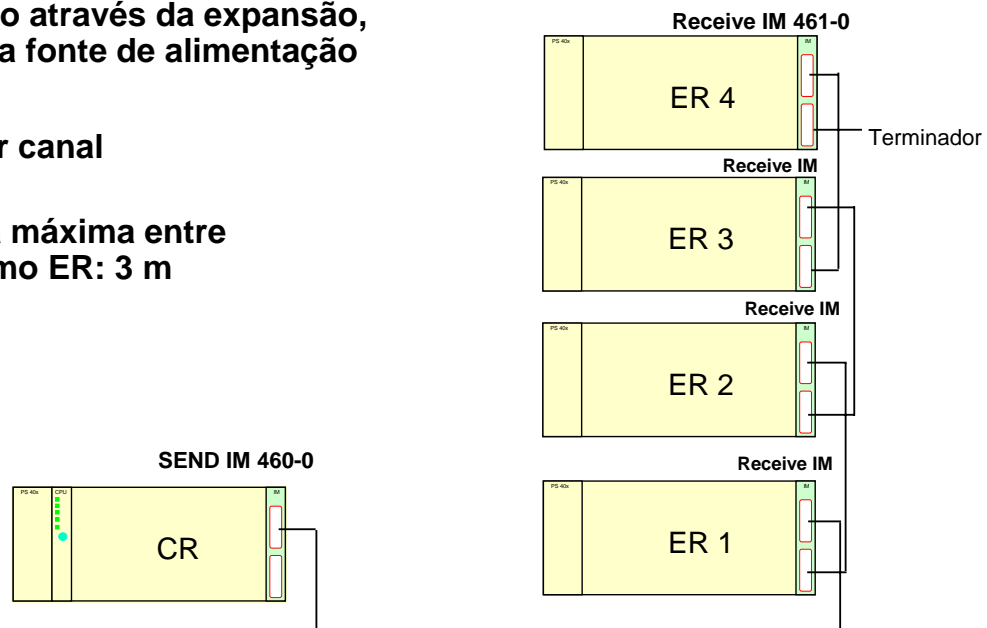
### Nota

Um máximo de 21 bastidores de expansão podem ser conectados a um bastidor central.

A IM Receive sempre deve ser inserido no slot mais a direita no bastidor de expansão.

## Expansão Centralizada 2

- ❑ Barramento P e barramento K conectado através da expansão, mas não a fonte de alimentação
- ❑ 4 ERs por canal
- ❑ Distância máxima entre CR e último ER: 3 m



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.12



Conhecimento em Automação  
Training Center

**Config. centralizada 2** Os módulos de interface IM 460-0/IM 461-0 são inseridos para a conexão centralizada dos bastidores de expansão para um bastidor central S7. Expansão centralizada tipo 2 tem as seguintes características:

- Um máximo de 8 bastidores de expansão podem ser conectados (4 por interface).
- A máxima distância entre o bastidor central e o último bastidor de expansão é de 3 m.
- Um máximo de 6 Send IM 460-0 podem ser inseridos por bastidor central.
- O módulo de interface Send IM 460-0 conecta o barramento P e o barramento K através do bastidor de expansão. Os módulos no bastidor de expansão não são alimentados com tensão através do cabo de conexão.

Por esta razão, cada bastidor de expansão deve ter seu próprio módulo de fonte de alimentação inserido.

- Um conector de interface não ocupado no Send IM 460-0 não tem que ser terminado; um conector de interface não ocupado no Receive IM 461-0 deve ser terminado com um terminador.
- Uma chave codificada com a qual o número do bastidor de expansão deve ser selecionado é localizada no Receive IM 461-0.

### Nota

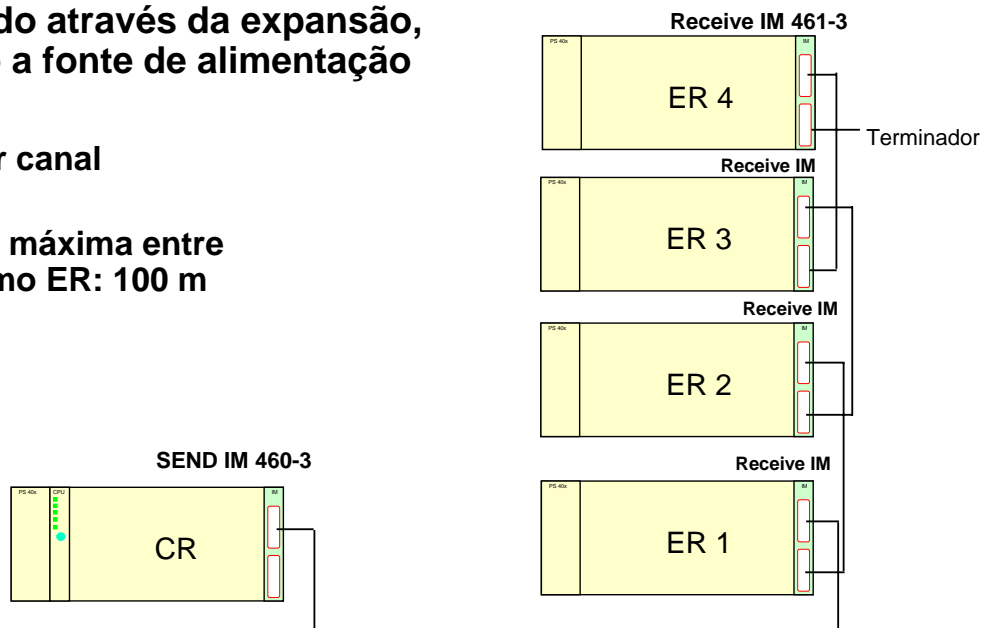
Um máximo de 21 bastidores de expansão podem ser conectado a um bastidor central.

O K barramento somente é conectado através do cabo aos 6 primeiros bastidores de expansão, isto é, módulos inteligentes como FMs e CPs podem desta forma somente ser operados nos 6 primeiros ERs.

O IM Receive deve sempre ser inserido no último slot direito do bastidor de expansão.

## Expansão Distribuída

- **Barramento P e barramento K conectado através da expansão, mas não a fonte de alimentação**
- **4 ERs por canal**
- **Distância máxima entre CR e último ER: 100 m**



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.13



Conhecimento em Automação  
Training Center

### Config. distribuída

Os módulos de interface IM 460-3/IM 461-3 são inseridos para a conexão distribuída dos bastidores de expansão para um bastidor central S7. Expansão distribuída tem as seguintes características:

- Um máximo de 8 bastidores de expansão podem ser conectados (4 por interface).
- A distância máxima entre o bastidor central e o último bastidor de expansão é de 100 m.
- Um máximo de 6 Send IM 460-3 podem ser inseridos por bastidor central.
- O módulo de interface Send IM 460-3 conecta o barramento P e o barramento K através do bastidor de expansão. Os módulos no bastidor de expansão não são alimentados com a tensão através do cabo de conexão.

Por esta razão, cada bastidor de expansão deve ter seu próprio módulo de fonte de alimentação inserido.

- Um conector de interface não ocupado no Send IM 460-3 não tem que ser terminado; um conector de interface não ocupado no Receive IM 461-3 deve ser terminado com um terminador.
- Uma chave codificada com a qual o número do bastidor de expansão deve ser selecionado é localizada no Receive IM 461-3.

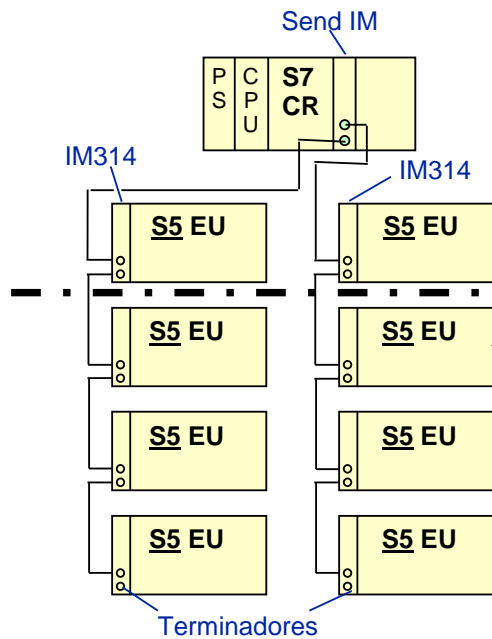
### Nota

Um máximo de 21 bastidores de expansão podem ser conectado a um bastidor central.

O K barramento somente é conectado através do cabo aos 6 primeiros bastidores de expansão, isto é, módulos inteligentes como FMs e CPs podem desta forma somente ser operados nos 6 primeiros ERs.

O IM Receive deve sempre ser inserido no último slot direito do bastidor de expansão.

## Conexão Distribuída entre S7 e S5



- Máx. 4 unidades expansão S5 por canal
- Máx. 4 IMs SEND no bastidor central
- Máx. distância do CR para último EU no canal: 600m
- Barramento S5 paralelo conectado através da expansão
- Possíveis unidades de expansão S5:  
EU 183 U, EU 185 U,  
ER 701-2, ER 701-3
- Outros EUs S5
- Máx. 32 EUs S5 por CR S7-400

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.14



Conhecimento em Automação  
Training Center

### Config. distribuída com EUs S5

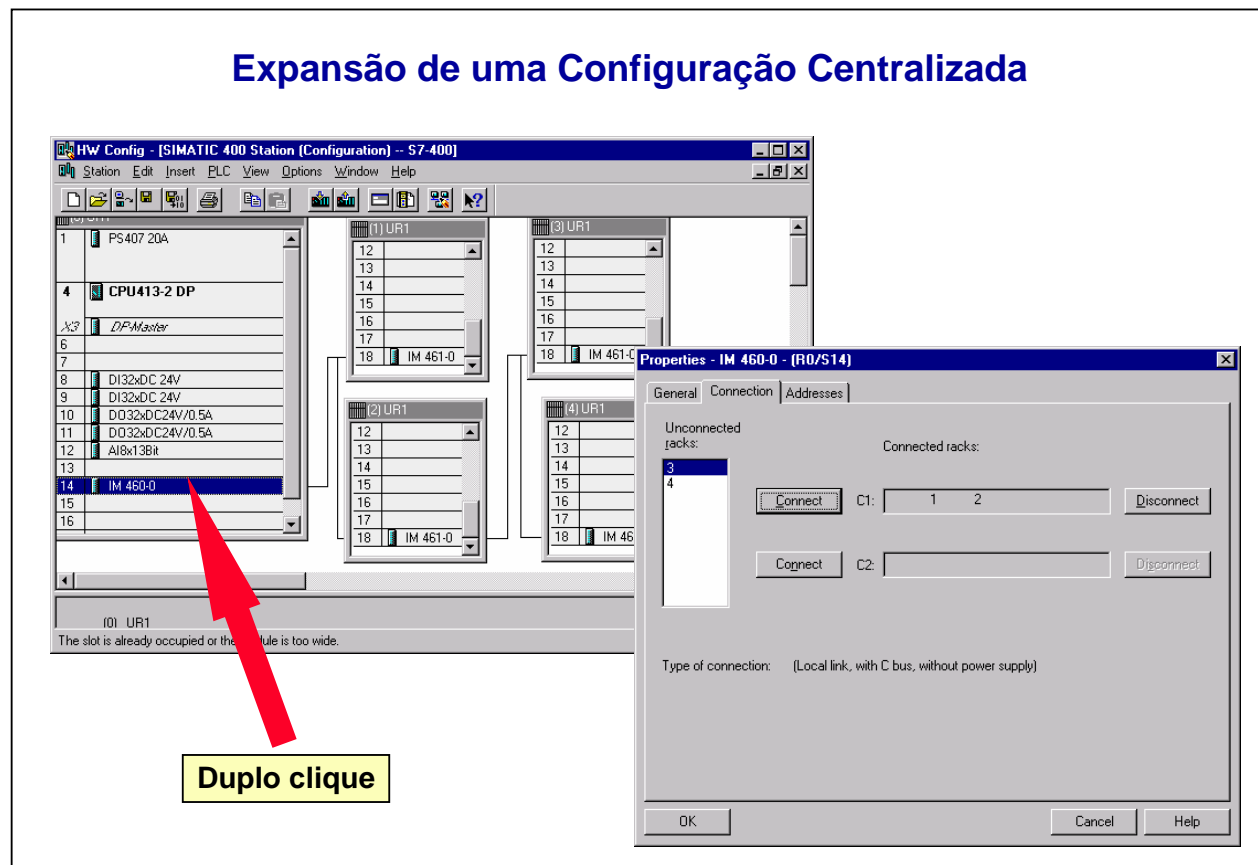
O módulo de interface IM463-2 habilita unidades de expansão S5 para ser conectado a um bastidor central S7-400.

As seguintes regras se aplicam a conexões com unidades de expansão S5 :

- Máx. 4 módulos de interface IM463-2 no bastidor central
- Máx. 4 unidades de expansão por canal
- Máx. distância entre o CR e o último EU: 600 m
- Os sistemas S5 podem ser conectados a um bastidor central S7-400 se o primeiro EU S5 é um EU-183U/EU-185U para o S5-135U/-155U ou um ER-701-2/ER-701-3 no caso de um S5-115U.

Os remanecentes EUs podem ser expandidos de acordo com as regras S5.

## Expansão de uma Configuração Centralizada



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.15



Conhecimento em Automação  
Training Center

### Expansão the configuração

Se você deseja “conectar” outros bastidores a um bastidor central e “equipá-los”, então proceda como a seguir:

1. Do catálogo de hardware selecione o bastidor (expansão)desejado.
2. Arraste os bastidores um após o outro para dentro da janela estação usando Marcar&Arrastar.
3. Insira os módulos desejados para dentro dos bastidores.

Importante: Os módulos de interface Receive devem ser inseridos em todos os bastidores de expansão, antes é possível uma conexão ao módulo de interface Send no bastidor central.

4. Somente para S7-400: De forma a estabelecer a conexão entre o IM Send e os IMs Recebe dos bastidores de expansão, proceda como a seguir:

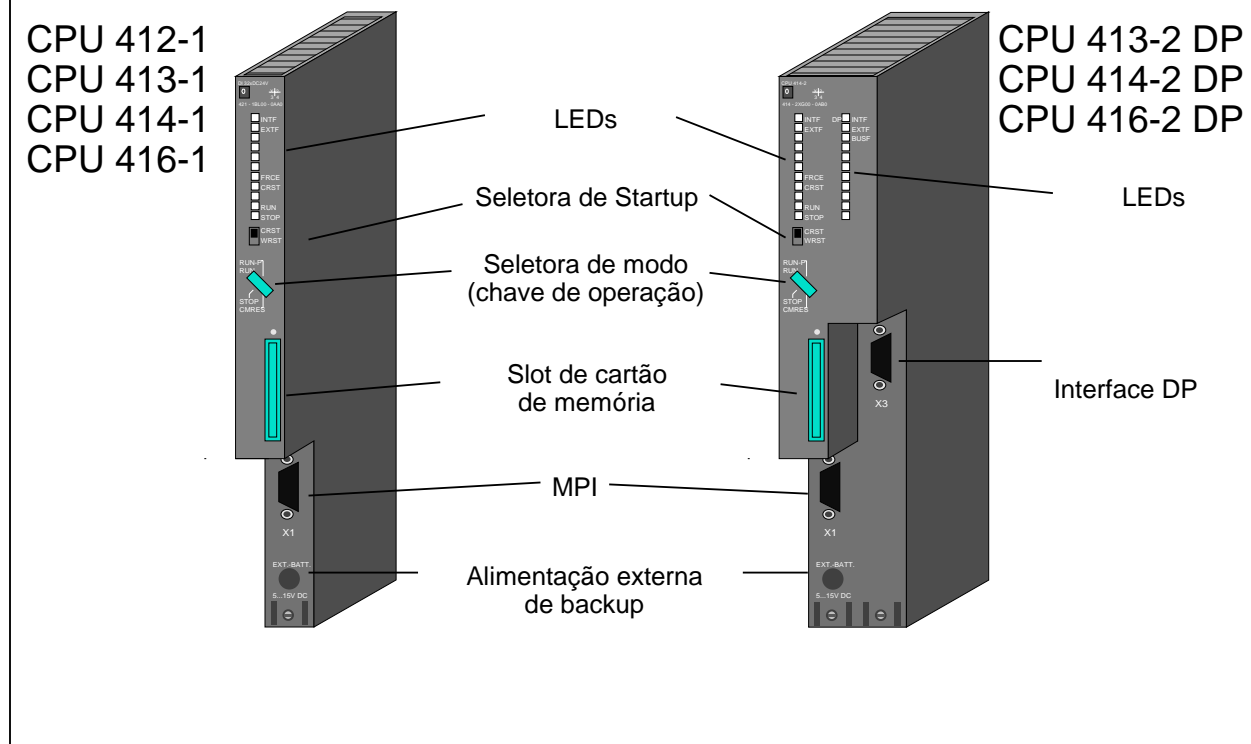
- Duplo clique no IM Send
- Selecione na tabela "conexão" todos os bastidores que não estão conectados são mostrados em uma lista.
- Selecione cada bastidor deste caso e usando o botão de comando “Connect” conecte-os às interfaces IM Send desejadas (C1 e C2).

Subseqüentemente, linhas de conexão entre os bastidores serão mostradas na janela estação.

### Peculiaridades CR2

Com o bastidor central CR2, você deve primeiro fazer a conexão entre os bastidores vazios (exceto para o IM Receive) e os IMs Send da respectiva CPU, antes você pode inserir módulos para dentro de um bastidor de expansão.

## Módulos CPU



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.16



Conhecimento em Automação  
Training Center

#### Conector p/Bateria

Tensão de alimentação externa de 5V a 15V DC para manter os dados (backup) na RAM, quando for trocado o módulo fonte de alimentação. A RAM pode ser mantida pela bateria interna do módulo fonte de alimentação ou através da conexão "EXT-BATT".

O conector é projetado para um plug jack de 2,5 mm.

#### Interface MPI

A interface MPI é utilizada para conectar equipamentos programáveis e/ou sistemas de interface com operador.

#### Interface integrada PROFIBUS DP

As CPUs 413-2/414-2/416-2 fornecem uma interface PROFIBUS-DP Mestre integrada para conexão de I/O distribuído, como ET200M, ET200U (B/C), S7-300, etc.

#### Cartões de memória

Cartões RAM ou FLASH-EPROM podem ser inseridos nas CPUs S7-400 como memória de carga externa de acordo com requisitos individuais.

- Cartões RAM com 64KB, 256KB, 1MB, 2MB de memória são mantidos seguros (backed up) através da bateria na unidade fonte de alimentação.
- Cartões FLASH-EPROM com 64KB, 256KB, 1MB, 2MB, 4MB, 8MB, 16MB.

#### Modos de Operação

MRES = Reset de memória.

STOP = Modo STOP, isto é, nenhum processamento de programa.

RUN = O programa é executado, acesso pela PG somente como read-only (somente leitura).

RUN-P = O programa é executado, acessos de leitura e escrita pela PG é possível.

#### Seletora de Startup

CRST= Um restart completo da CPU (Cold Restart) é realizado quando a CPU é inicializada com a chave seletora está neste modo.

WRST= Um restart da CPU (Warm Restart) é realizado quando a CPU é inicializada com a chave seletora está neste modo.

## Dados Técnicos das CPUs S7-400 (1)

CPU	412-1	413-1	413-2 DP	414-1	414-2 DP	416-1	416-2 DP	417-4
<b>Tempo de execução por instrução binária</b>	200 ns	200 ns	200 ns	100 ns	100 ns	80 ns	80 ns	100 ns
<b>Carga/Transferência</b>	200 ns	200 ns	200 nsc	100 ns	100 ns	80 ns	80 ns	100 ns
<b>Ponto fixo 16-bit (+/-)</b>	200 ns	200 ns	200 ns	100 ns	100 ns	80 ns	80 ns	100 ns
<b>Pto. flut. IEEE (+/-)</b>	1,2 µs	1,2 µs	1,2 µs	0,6 µs	0,6 µs	0,48 µs	0,48 µs	0,48 µs
<b>Memória usuário</b>								
Memória Trabalho	48 KB	72 KB	72 KB	128 KB	128/384 KB	512 KB	0.8/1.6 MB	4...20 MB
Memória carga integ.	8 KB	8 KB	8 KB	8 KB	8 KB	16 KB	16 KB	256 KB
Memória carga ext.	15 MB	15 MB	15 MB	15 MB	15 MB	15 MB	15 MB	64 MB
<b>Endereços</b>								
Memória Bit	4096	4096	4096	8192	8192	16384	16384	16384
Memória pulsos	8	8	8	8	8	8	8	8
Temporizadores	256	256	256	256	256	512	512	512
Contadores	256	256	256	256	256	512	512	512
<b>Tipo/número Bloco</b>								
FBs	256	256	256	512	512	2048	2048	6144
FCs	256	256	256	1024	1024	2048	2048	6144
DB's	511	511	511	1023	1023	4095	4095	8192
<b>Tam. imagem processo (tabelas entr./saída)</b>	128 bytes cada	128 bytes cada	128 bytes cada	256 bytes cada	256 bytes cada	512 bytes cada	512 bytes cada	1024 bytes cada
<b>Máx. espaço ender. I/O</b>	0.5 KB <sup>1)</sup> cada	1 KB <sup>1)</sup> cada	1 KB <sup>1)</sup> cada	2 KB <sup>1)</sup> cada	4 KB <sup>1)</sup> cada	4 KB <sup>1)</sup> cada	8 KB <sup>1)</sup> cada	16 KB <sup>1)</sup> cada
<b>Interfaces integradas</b>	MPI	MPI	MPI, DP	MPI	MPI, DP	MPI	MPI, DP	MPI, 4 x DP

<sup>1)</sup> 1 Byte = 8 entradas / saídas digitais  
2 Bytes = 1 entrada / saída analógica

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.17



Conhecimento em Automação  
Training Center

### Tipos de CPUs

Para cada faixa de performance existe uma CPU correspondente com o correspondente tempo de execução, tamanho da memória de trabalho e número de blocos.

### Programação

Programas do usuário são escritos em concordância com IEC 1131-3. As seguintes novas características estão integradas no S7:

- Instruções para processamento de valor analógico:
  - Aritmética de ponto fixo e ponto flutuante (32-bit)
  - Raiz quadrada e elevado ao quadrado
  - Funções logarítmicas
  - Funções trigonométricas
- Novos tipos de dados para programação orientada ao problema, por exemplo:
  - ARRAY
  - STRUCT (estrutura)
  - POINTER
- Conceito de bloco orientada a objeto, por exemplo:
  - FBs com bloco de dados instance
  - Modelo multi-instance
- Blocos do sistema integrados, por exemplo para comunicação, etc.

### I/O de processo

Os endereços lógicos dos módulos de I/O são arranjados em um espaço de endereço linear de tamanho conveniente.

Os endereços de estações escravas conectadas à interface DP integrada são também mapeadas neste espaço de endereço linear. Deste modo, o I/O distribuído pode ser endereçada da mesma maneira que o I/O central no programa do usuário.

O STEP 7 é utilizado para atribuir parâmetros de endereço para a central e o I/O distribuído.

## Dados Técnicos das CPUs S7-400(2)

CPU	412-1	413-1	413-2 DP	414-1	414-2 DP	416-1	416-2 DP	417-4
<b>Blocos de organização</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>	<b>OB No.</b>
Ciclo livre	1	1	1	1	1	1	1	1
Interrup. horário do dia	10,11	10,11	10,11	10-13	10-13	10-17	10-17	10-17
Interrup. atraso tempo	20,21	20,21	20,21	20-23	20-23	20-23	20-23	20-23
Interrupções cíclicas	32,35	32,35	32,35	32-35	32-35	30-38	30-38	30-38
Interrup. de hardware	40,41	40,41	40,41	40-43	40-43	40-47	40-47	40-47
Inter. multiprocessam.	60	60	60	60	60	60	60	60
Background	90	90	90	90	90	90	90	90
Startup	100-102	100-101	100-102	100-102	100-101	100-102	100-102	100-102
Erros, síncronos	80-87	80-87	80-87	80-87	80-87	80-87	80-87	80-87
Erros, assíncronos	121,122	121,122	121,122	121,122	121,122	121,122	121,122	121,122
<b>Dados Locais</b>	<b>4 KB</b>	<b>4 KB</b>	<b>4 KB</b>	<b>8 KB</b>	<b>8 KB</b>	<b>16 KB</b>	<b>16 KB</b>	<b>24 KB</b>
<b>Máx. comprim. bloco</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>	<b>64 KB</b>
<b>Tamanho aninhamento de bloco por classe</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>16</b>	<b>24</b>
<b>Comunicação controlada por programa:</b>								
<b>Máx. No. de conexões</b>	<b>8</b>	<b>16</b>	<b>16</b>	<b>32</b>	<b>32</b>	<b>64</b>	<b>64</b>	<b>64</b>
<b>Comunicação de Dados Globais através de MPI:</b>								
<b>Círculos GD por CPU</b>	<b>8</b>	<b>8</b>	<b>8</b>	<b>8</b>	<b>8</b>	<b>16</b>	<b>16</b>	<b>16</b>
<b>Envio de pacotes GD por círculos GD</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>	<b>1</b>
<b>Recebimento de pacotes GD por círculos GD</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>2</b>
<b>Máx. dados de usuário por pacote</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>	<b>54 bytes</b>

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.18Conhecimento em Automação  
Training Center

### Comunicação

O S7-400 fornece diversas facilidades diferentes de comunicação.

1. Interface integrada multiponto (interface MPI), na qual PGs/PCs, sistemas IHM, sistemas M7-300/400 e demais sistemas S7-300/400 podem ser conectados como nós ativos.
2. Interfaces PROFIBUS DP integradas nas CPUs 413-2 / 414-2 / 416-2 / 417-4. para conexão a um sistema I/O distribuído (isto é, ET200) para a CPU.
3. Processadores de comunicação como a CP443 para conexão aos sistemas de barramento PROFIBUS e Ethernet Industrial.
4. Processadores de comunicação como a CP441 para poderosa comunicação ponto a ponto (PtP) para outros S7, S5 ou controladores e sistemas externos.

### Funções S7

As funções de comunicação S7 são divididas em dois tipos:

Comunicação S7 básica: pequenos volumes de dados (até 76 bytes) podem ser trocados entre parceiros de comunicação (S7-300/400) através de MPI ou com uma estação (e com escravos inteligentes através de PROFIBUS DP) com este serviço.

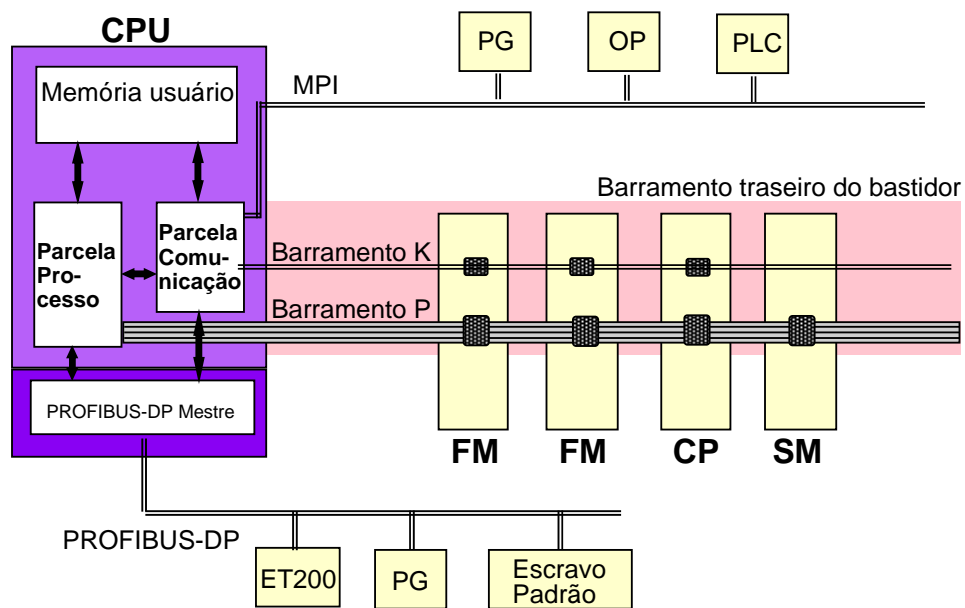
As SFCs de comunicação associadas estão integradas no sistema operacional. Eles não requerem conexões configuradas, a alocação de recursos de comunicação e o endereçamento do parceiro de comunicação ocorre diretamente na chamada da SFC.

Comunicação S7 estendida: grandes volumes de dados (até 64 Kbytes) podem ser trocados independentemente de uma rede de comunicação (MPI, Profibus ou Ethernet Industrial) com este serviço.

Os SFBs associados estão integrados no sistema operacional para S7-400 (não para S7-300, S7-300 é somente server) e requer uma conexão configurada na chamada do SFB. Conexões configuradas são estabelecidas de acordo com a tabela de conexão quando o sistema é inicializado e os recursos relevantes são atribuídos estaticamente.



## Arquitetura do Sistema



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.19



Conhecimento em Automação  
Training Center

**Configuração da CPU** Devido ao tipo de meio de comunicação, em função da configuração dos sistemas distribuídos, as CPUs S7-400 são divididas em duas unidades funcionais:

- Parçela processo
- Parçela comunicação

### Barramento P

A parçela processo da CPU permite acesso aos módulos de sinal através do barramento P. Ele é otimizado para a troca de até 4 bytes de dados.

O barramento P no sistema S7-400 tem as seguintes características:

- 8 bits de largura
- paralelo
- tempo de acesso de 1,5 µs

### Barramento K

O barramento K (barramento de comunicação) realiza a troca de dados assíncronos com módulos inteligentes, capazes de comunicação, isto é, um FM ou um CP. Ele é otimizado para a troca de grandes quantidades de dados.

O barramento K é projetado como um barramento multi-mestre; ele é uma extensão lógica da interface MPI. Ele tem as seguintes características:

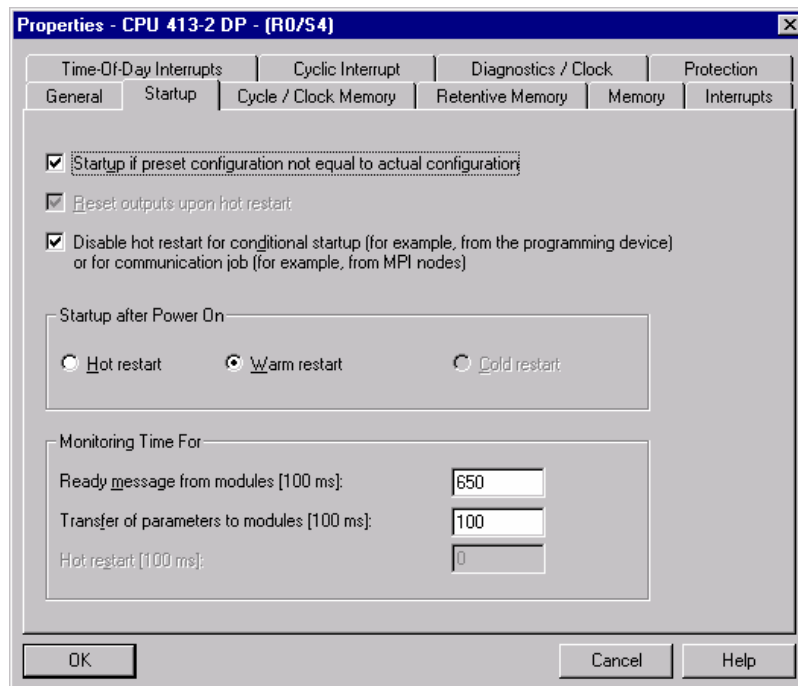
- serial
- baud rate: 10,5 MB/seg
- máx. 127 nós (teoricamente)

### MPI

Comunicação através da interface MPI tem as seguintes características :

- serial
- baud rate: 187,5 KB/seg
- máx. 32 nós

## Parâmetros da CPU: Características de Startup



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.20



Conhecimento em Automação  
Training Center

#### Configuração Setpoint / Atual

Através da desativação deste campo você pode, no S7, fazer com que a CPU vá para Stop após startup se a configuração atual não for igual a configuração setpoint (não estiver de acordo com a configuração).

#### Teste de Hardware

Através da ativação desta função, a RAM interna da CPU é testada durante o startup. A duração do teste depende do tamanho da memória.

#### Deletar PIQ..

No restart de um S7-400, a imagem de processo é deletada pelo padrão após o ciclo de varredura remanescente ser processado. Se esta característica não for desejada, você pode deselecioná-la.

**Desabilitação Restart** Restrição para um Restart completo para o startup manual do S7-400.

#### POWER ON

Com o S7-400 você pode escolher entre:

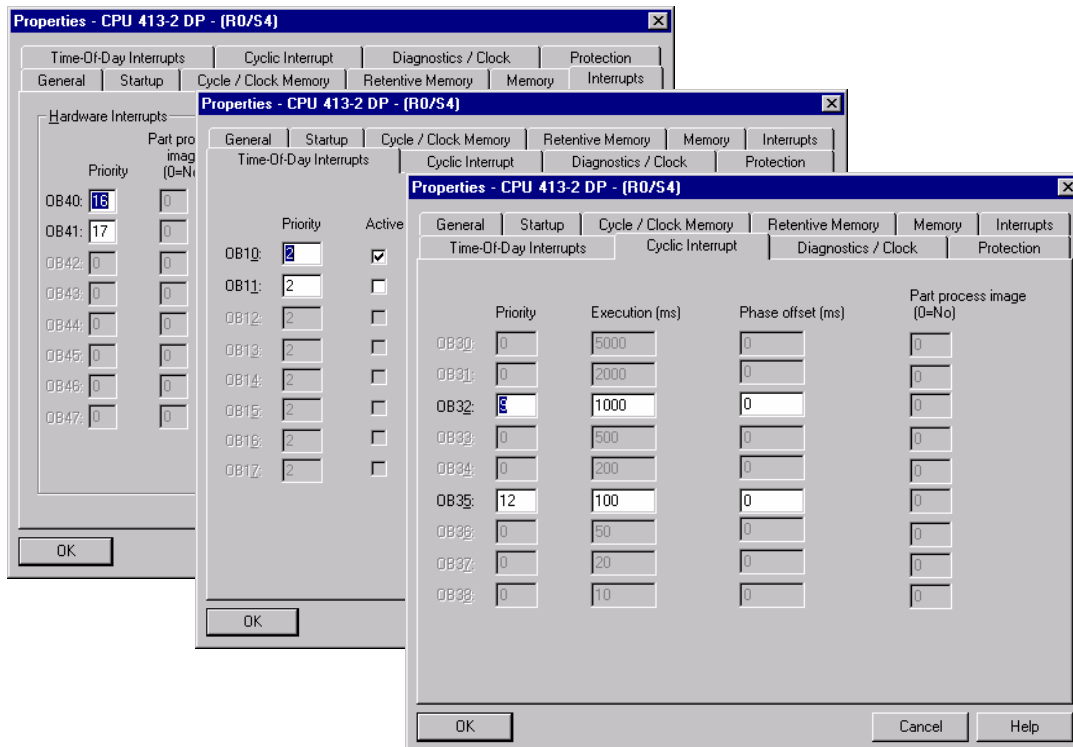
- Restart completo (apaga as áreas não retentivas e o processamento do programa começa com a primeira instrução no bloco OB1).
- Restart (todas as áreas de memória são mantidas intactas e o programa continua do ponto em que foi interrompido).
- Restart a frio (Cold restart) (apaga as áreas retentivas e não retentivas e o processamento do programa começa com a primeira instrução no bloco OB1).

#### Tempos de Monitoração

Os seguintes tempos podem ser especificados:

- Máximo tempo de espera até que todos os módulos tenham se reportado à CPU.
- Máximo tempo até que um módulo deva ter reconhecido uma transferência de parâmetro.
- Para o S7-400, o máximo tempo após uma falha de alimentação, após a qual um restart possa ser realizado.

## Parâmetros da CPU: Interrupções



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.21Conhecimento em Automação  
Training Center

### Prioridades

No S7-400 você pode alterar as prioridades dos blocos de manipulação de interrupção e assim estabelecer a ordem na qual elas serão processadas quando diversos eventos de interrupção ocorrerem simultaneamente. As prioridades são de 1 a 24 e a interrupção com a maior prioridade será processada primeiro.

### Hardware interrupções

Neste parâmetro de bloco você seta as prioridades os blocos de organização para interrupções de hardware. Você pode setar as prioridades 0 e de 2 a 24 (0 deseleciona).

### Interrupção Horário do dia (Time-of-day)

Você pode usar estas interrupções para setar o tempo de partida para execução do programa somente uma vez ou para uma ativação a ser repetida Regularmene deste horário para frente (a cada minuto, a cada hora, diariamente, semanalmente, mensalmente, anualmente).

### Interrupção Cíclica

A interrupção cíclica pode ser utilizada para executar uma parte de um programa em intervalos fixos. Você pode selecionar intervalos de 1 a 60000 ms. Isto habilita você, por exemplo, a implementar tarefas de controle em malha fechada que devam ser processadas em intervalos fixos de tempo (por amostragem de tempo).

No S7-400 existem oito diferentes ciclos de interrupções com diferentes intervalos. Para evitar que todas as interrupções cíclicas venham a ocorrer ao mesmo tempo, você pode ajustar o "Phase Offset" para que as chamadas das interrupções cíclicas não ocorram aos mesmo tempo.

### Interrupções Atraso de tempo

Uma interrupção atraso de tempo (time-delay) é uma chamada de um bloco de organização que ocorre uma única vez, o qual é ativada com um atraso de tempo, por exemplo após a recepção de um sinal de processo.

Interrupções atraso de tempo são manipuladas no programa do usuário com a ajuda das SFCs 32 a 34.

- SFC32 "SRT\_DINT" = Inicia uma interrupção atraso de tempo.
- SFC33 "CAN\_DINT" = Cancela interrupção atraso de tempo.
- SFC34 "QRY\_DINT" = Solicita o status da interrupção atraso de tempo.

## Parâmetros da CPU: Dados Locais

**Properties - CPU 413-2 DP - (R0/S4)**

Time-Of-Day Interrupts | Cyclic Interrupt | Diagnostics / Clock | Protection  
 General | Startup | Cycle / Clock Memory | Retentive Memory | Memory | Interrupts

Local Data (Priority Classes)

1	512	7	0	13	0	19	0	25	256
2	256	8	0	14	0	20	0	26	256
3	256	9	256	15	0	21	0	27	256
4	256	10	0	16	256	22	0	28	256
5	0	11	0	17	256	23	0	29	256
6	0	12	256	18	0	24	256		

Occupied 3840 Bytes of max. 2048

Communication Resources  
 Max. number of communication jobs: 150

OK Cancel Help

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
 File: PRO2\_11P.22



Conhecimento em Automação  
 Training Center

### Dados Locais

A tabela acima permite ao usuário definir as condições dos dados locais para cada classe de prioridade (OB).

Se a área de dados locais de uma classe de prioridade for excedida, o sistema é levado para o modo STOP.

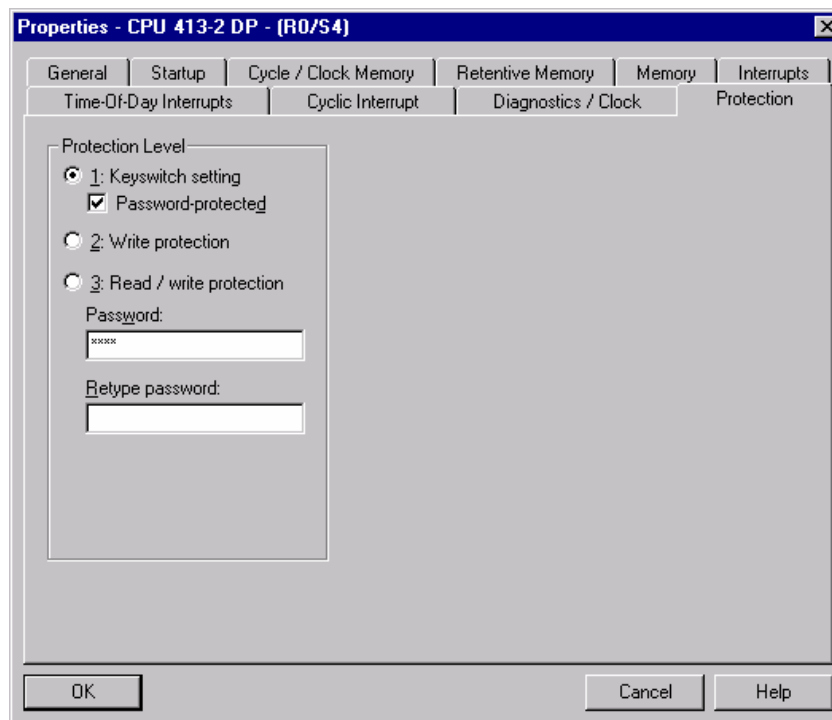
Se você deseja endereçar os dados locais simbolicamente, o editor STL/LAD/FBD assegura endereçamento e administração corretos.

### Tamanho do L Stack

A quantidade de dados locais depende da CPU.

- CPU 412 - 4 KB de dados locais
- CPU 413 - 4 KB de dados locais
- CPU 414 - 8 KB de dados locais
- CPU 416 - 16 KB de dados locais
- CPU 417 - 32 KB de dados locais

## Parâmetros da CPU: Conceitos de Proteção



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.23



Conhecimento em Automação  
Training Center

### Função

Neste diálogo você pode selecionar um dos três níveis de proteção, de forma a proteger a CPU de acessos não autorizados.

### Ajustes das Características

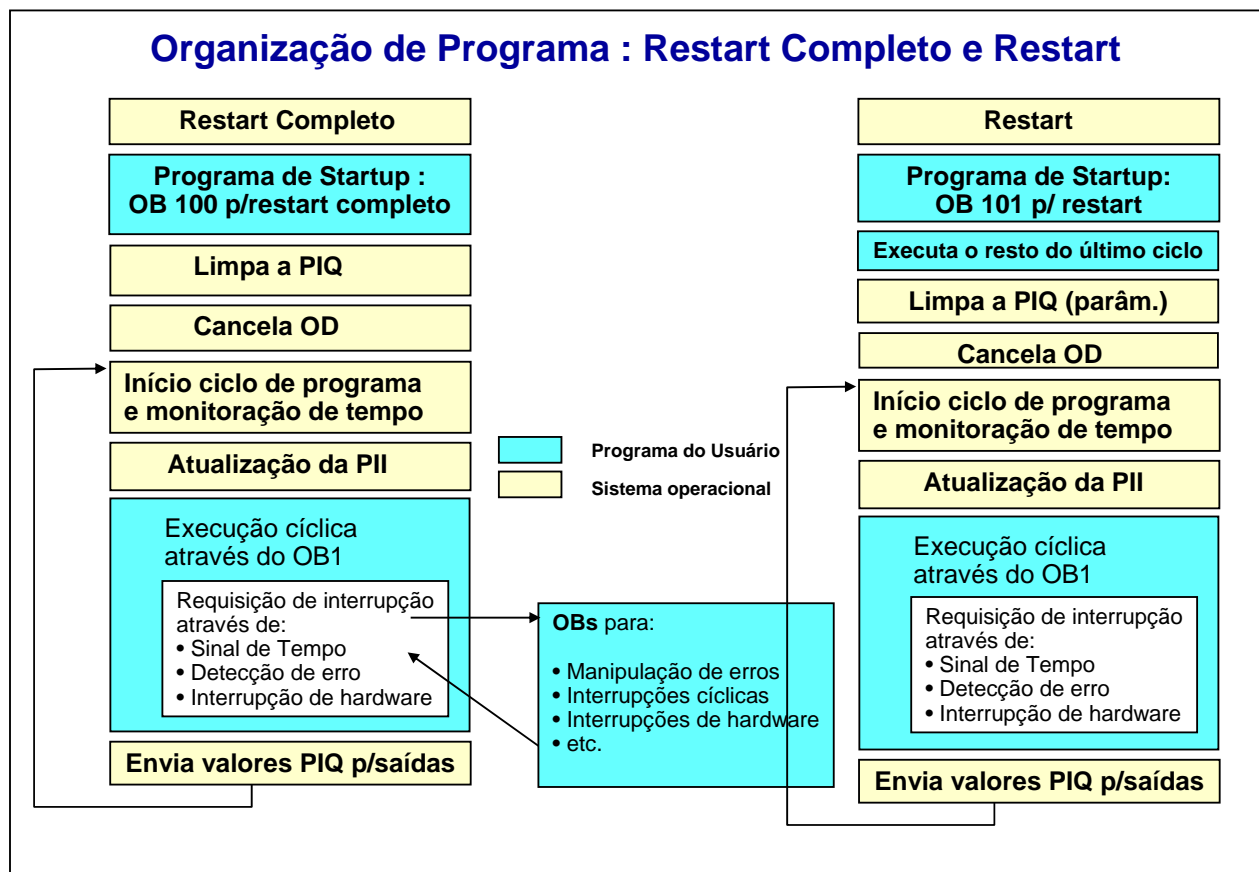
O nível de proteção 1 (nenhum password parametrizado): A posição da chave seletora do modo de operação da CPU determina a proteção:

- chave na posição RUN-P ou STOP: sem restrições de acesso
- chave na posição RUN: é possível somente o acesso a leitura!

### Níveis de Proteção Parametrizados

Se você tiver parametrizado um nível de proteção com password:

- acessos de leitura e escrita são possíveis quando o password é conhecido, independente da posição da chave seletora e independente do nível de proteção parametrizado.
- as seguintes restrições são aplicáveis quando o password não é conhecido:
  - Nível de Proteção 1: corresponde ao descrito no parágrafo acima (Ajustes de Características).
  - Nível de Proteção 2: é permitido somente o acesso a leitura, independente da posição da chave seletora do modo de operação da CPU.
  - Nível de Proteção 3: nenhum acesso, seja de leitura seja de escrita é possível, independente da posição da chave seletora.



## SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.24Conhecimento em Automação  
Training Center**Restart Completo**

No restart completo, as áreas de memória não retentivas das memórias bit, contadores e temporizadores são resetas após o OB100 ter sido processado. OB1 sempre inicia com a primeira instrução.

**Cold Restart  
(restart a frio)**

As ações que ocorrem durante um restart a frio são similares aquelas para um restart completo, com as seguintes diferenças:

- O OB102 é chamado em vez do OB100.
- Os blocos de dados gerados pelos SFCs durante a execução cíclica são apagados, os outros blocos de dados são inicializados com valor da memória de carga.
- A imagem de processo e todos os temporizadores, contadores e memórias de bit are resetados, se eles tiverem sido parametrizados como retentivos ou não.

**Restart**

No restart, após o OB101 de restart ter sido executado, a execução do OB1 continua do ponto onde ele foi interrompido, isto é o restante do ciclo é completado com as memórias bit retentivas, temporizadores e contadores.

**OD**

Output disable (desabilitação da saída), cabo de barramento no S7 (corresponde ao comando inibir saída no S5).

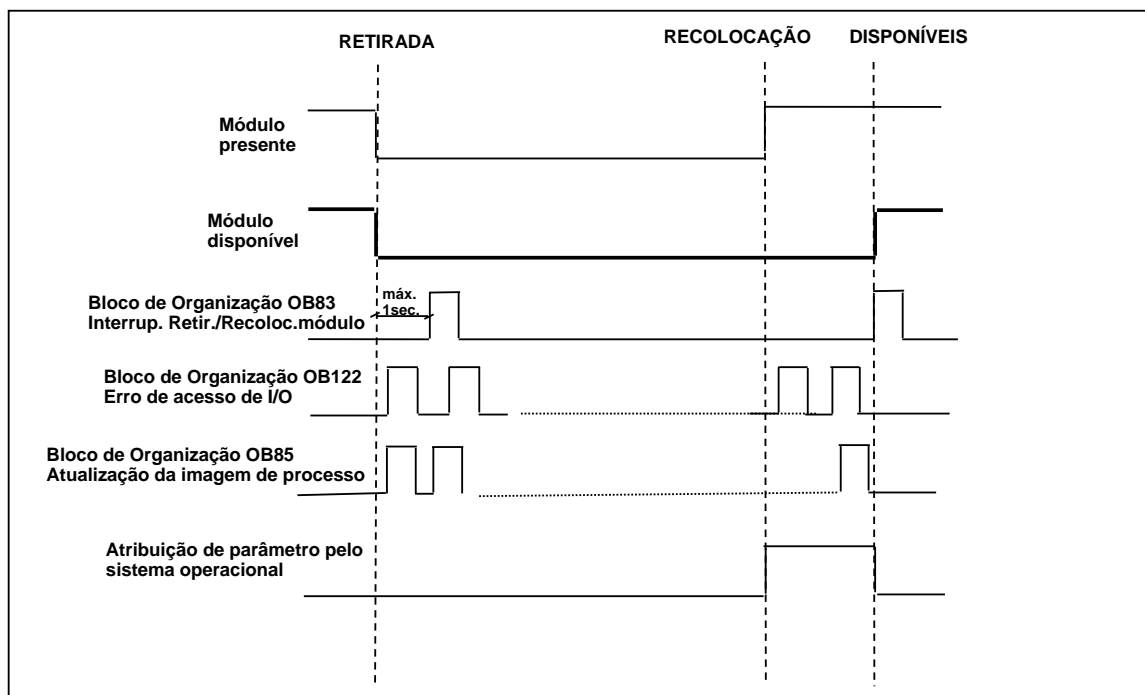
**Ações**

O sistema operacional realiza as seguintes ações na partida (startup):

- Limpa as pilhas (stacks) (C/CR)
- Reseta as memórias bit não retentivas, temporizadores e contadores (CR)
- Reseta todas as memórias bit, temporizadores e contadores (C)
- Limpa imagem de processo de saída PIQ (C/CR), se parametrizada (R)
- Reseta área de memória de saída (C, CR), se parametrizada (R)
- Limpa interrupções (C/CR/R) através da OD
- Atualiza a lista de estados do sistema (C/CR/R)
- Transfere configuração para os módulos (C/CR/R)

(CR= Restart Completo, C=Cold Restart, R= Restart).

## A Interrupção de Inserção/Remoção de Módulos no S7 - 400



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.25



Conhecimento em Automação  
Training Center

### OB83 interrupção Retirada e Recoloc.

No S7-400 você pode retirar e recolocar módulos, enquanto está alimentado, em modo RUN ou em STOP. As exceções são CPUs, fontes de alimentação, módulos S5 em adaptadores de módulos e IMs.

Após retirada de um módulo autorizado em modo RUN, o sistema operacional da CPU pode chamar um dos seguintes blocos de organização, dependendo da situação:

- OB85 – atualização da imagem de processo
- OB122 – I/O erro de acesso
- OB83 – evento Retirada&Recolocação.

O usuário deve tomar em consideração que o OB83 somente é chamado após aproximadamente 1segundo, enquanto outros OBs, como uma regra, são ativados brevemente.

Após recolocação de um módulo, ele é verificado pela CPU e – se nenhum tipo de erro existir – são atribuídos parâmetros. Após uma atribuição ordenada de parâmetros, o módulo está disponível para uso.

Se um erro é reconhecido na atribuição de parâmetros, o OB82 diagnostica a interrupção é automaticamente iniciado.

### Informação de Partida no OB83

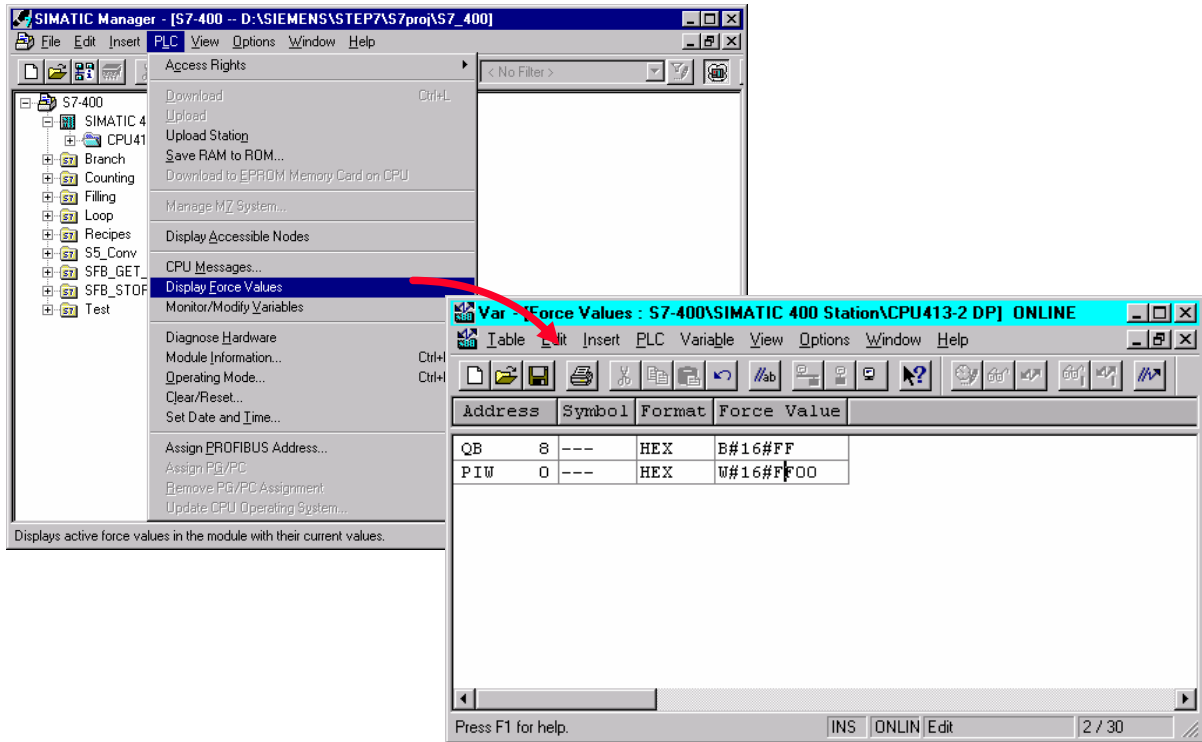
As seguintes informações existem nos dados locais do OB83:

- Retirada/Recolocação de um módulo
- Endereço lógico do módulo
- Tipo atual do módulo

### Replacement Value

Você pode usar as funções do sistema SFC 44 (RPL\_VAL) para substituir um valor recolocado para os sinais de processo perdido de um módulo de entrada em um OB de erro.

## O Comando Force no S7-400



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.26



Conhecimento em Automação  
Training Center

### Forçando

Com a função *Forcing* você pode, no S7-400, estabelecer valores pré definidos para variáveis do programa do usuário.

### Notas para Forcing

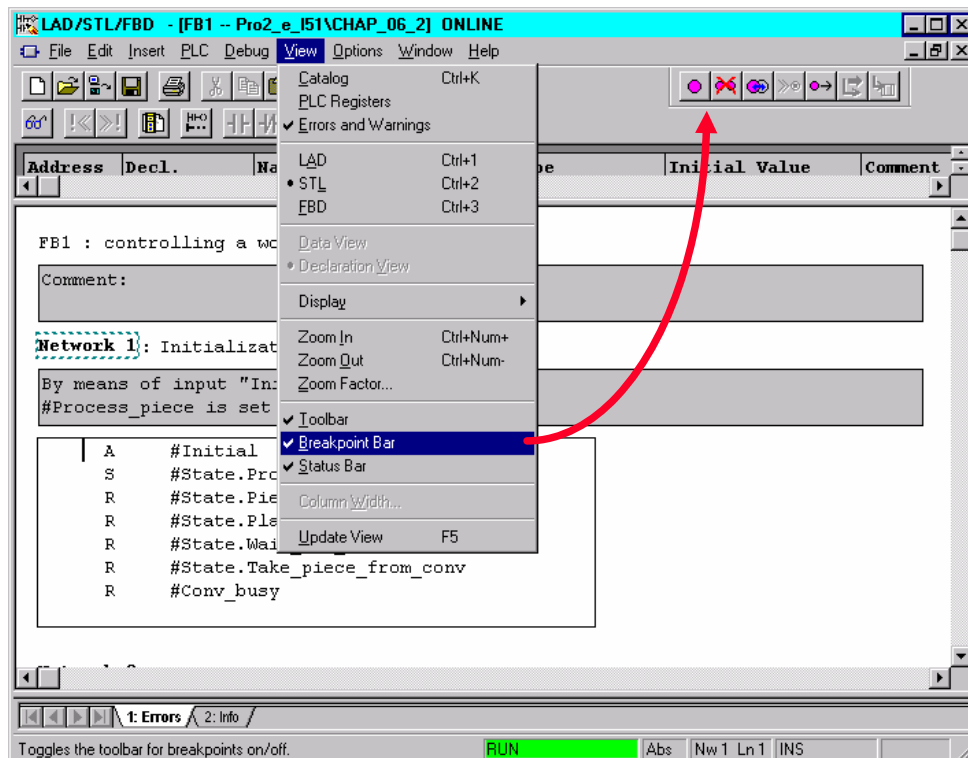
- Antes de você iniciar a função "Force", você deve ter certeza que ninguém mais está realizando a mesma função ao mesmo tempo na mesma CPU.
- Uma tarefa force somente pode ser deletada ou terminada com o comando de menu *Variable -> Stop Forcing*.
- "Forcing" não pode ser desfeita com o comando de menu *Edit -> Undo*.
- O fechamento da janela Force Values ou o término da aplicação "Monitor/Modify Variables" não apaga a tarefa force.
- Ler as informações fornecidas pela ajuda On-line (Help) sobre as diferenças entre forçar variáveis e modificar variáveis.

### Selecionando a Função "Force"

1. Dentro do gerenciador SIMATIC selecione a CPU a ser forçada e então use o comando de menu *PLC -> Display Force Values* para abrir a janela Force Values na qual o estado lógico atual da CPU selecionada é mostrado. Somente quando a janela "Force Values" está ativa é possível selecionar o comando de menu para forçar. Se nenhuma tarefa de force está correntemente ativa, a janela fica vazia. Se uma tarefa de force está já ativa, as variáveis são mostradas em negrito com o correspondentes valores forçados.
2. Na coluna de "endereço", insira as variáveis que você deseja forçar. Na coluna "Force Value", insira os valores que você deseja atribuir para as variáveis.
3. Iniciar a tarefa forçar com comando de menu *Variable -> Force*. Se nenhuma tarefa forçar estiver correntemente ativa, as variáveis são atribuídos os valores forçados.
4. Você pode terminar a tarefa forçar com o comando de menu *Variable -> Stop Forcing*.



## Ativando a Barra Breakpoint



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.27



Conhecimento em Automação  
Training Center

### Breakpoints (pontos de parada)

Com a ajuda desta função de teste é possível testar os programas gerados em representação STL no modo passo simples e desta forma estar habilitado a seguir a seqüência das instruções a serem executadas bem como verificar o comportamento dos registros associados.

Diversos breakpoints (pontos de parada) podem ser setados em um bloco. A quantidade de números de pontos de parada (breakpoints) possíveis depende do tipo de CPU:

- CPU 412, 413: 2 pontos de parada
- CPU 414: 3 pontos de parada
- CPU 416: 4 pontos de parada

### Notas

- O bloco deve ser aberto "on-line" de forma a selecionar a função "Breakpoint".
- A função "Breakpoints" somente pode ser selecionadas, se no diálogo *Debug -> Operation -> Test Operation* foi selecionado.
- O comando de menu *Execute Next Statement* ou *Execute Call* requer um breakpoint livre para a implementação interna.
- Quando um breakpoint se encontra no programa em processamento, a CPU muda do modo RUN para o modo HOLD. Neste modo o led STOP fica aceso e ao mesmo tempo o led RUN fica piscando.

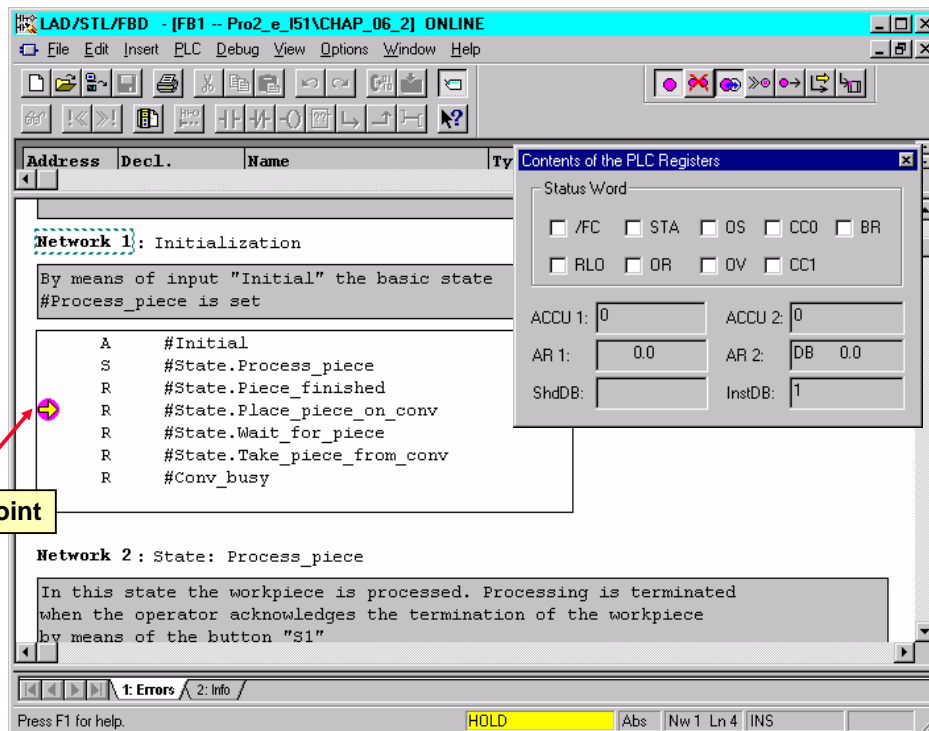
### Funções Breakpoint

As funções breakpoint podem ser selecionadas no editor de programas através do item de menu "Debug" ou da barra de ferramentas breakpoint.

### Barra Breakpoint

Você ativa a barra de ferramentas breakpoint no editor de programas através da opção de menu *View -> Breakpoint Bar*.

## Execução de Programas com Breakpoints (somente S7-400)



### SIMATIC S7

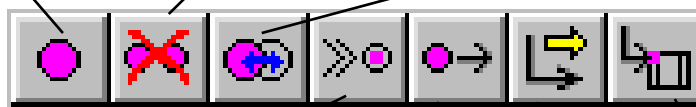
Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.28Conhecimento em Automação  
Training Center

### Barra Breakpoint

A barra breakpoint oferece os botões de comando para "Modo de Teste em Passo Simples".

Ativa Breakpoint    Deleta Breakpoints    Ativa/desativa Breakpoints



Mostra próximo Breakpoint    Continua    Próxima Instrução    Executa chamada

### Ativa Breakpoint

Com "Set Breakpoint" você determina em qual local do programa irá ocorrer a parada de execução do programa. A instrução no ponto de parada (breakpoint) não é executada.

### Deleta Breakpoint

Todos os breakpoints são deletados.

### Ativa / desativa Breakpoint

Com "Breakpoint Active" você ativa todos os breakpoints, não somente aqueles já setados mas também aqueles que estão para serem setados.

### Mostra próximo Breakpoint

Com "Show Next Breakpoint" o editor salta para o próximo breakpoint marcado, sem o programa ser processado.

### Continua

Com "Continue" o programa é executado até o próximo breakpoint ativo.

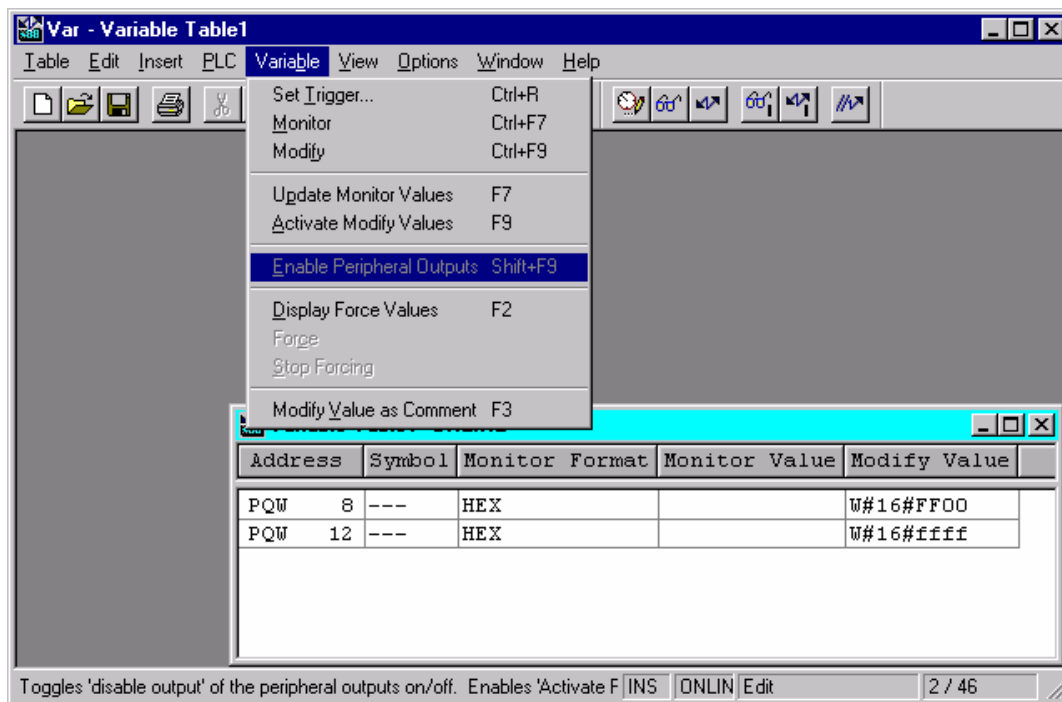
### Próxima Instrução

Com "Execute Next Statement" você processa o programa em modo passo simples. Se um bloco chamado é alcançado, você salta com "Execute Next Statement" para a primeira instrução após o bloco chamado.

### Executa Chamada

Quando a chamada de um bloco é alcançada, você salta para dentro do bloco com "Execute Call". No fim do bloco ocorre um salto de volta para a próxima instrução após a chamada do bloco.

## Habilitação de Saídas de Periferia (somente S7-400)



### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.29



Conhecimento em Automação  
Training Center

### Introdução

A função "enable peripheral outputs" (habilita saídas de periferia) desabilita ou desliga as saídas do CLP (peripheral outputs -PQ). Isto habilita você a modificar as saídas de periferia quando a CPU está em modo STOP.

### Convocação

Para habilitar as saídas de periferia, proceda como a seguir:

1. Use o comando de menu *Table -> Open* para abrir a tabela de variáveis (VAT) que contem as saídas de periferia que você deseja modificar ou ative a janela para a tabela de variáveis correspondente.
2. Selecione o comando de menu *PLC -> Connect To* para estabelecer uma conexão com a CPU desejada, então você pode modificar as saídas de periferia da tabela de variáveis ativa.
3. Abra o diálogo "Operation Mode" com o comando de menu *PLC -> Operation Mode* e mude o modo de operação da CPU para STOP.
4. Insira os valores apropriados para saídas de periferia que você deseja modificar na coluna "Modify Value".  
Exemplos: PQB 7 Modify value: 2#0001000011  
PQW 2 W#16#0027  
PQD 4 DW#16#0001
5. Use o comando de menu *Variable -> Enable Peripheral Output* para mudar para o modo "Enable Peripheral Output".
6. Use o comando de menu *Variable -> Activate Modify Values* para modificar as saídas de periferia. "Enable Peripheral Output" se mantém ativa até você selecionar o comando de menu *Variable -> Enable Peripheral Output* novamente para desativar esta função.
7. Para atribuir novos valores, reinicie no passo 4.

### Notas

- Se a CPU muda seu modo de operação e vai de STOP para RUN ou STARTUP, por exemplo, uma mensagem é mostrada.
- Se a CPU está em modo RUN e a função "enable peripheral outputs" é selecionada, uma mensagem também é mostrada.

## CP 441 para Conexões Ponto-a-Ponto

### Interfaces:

- CP 441-1: 1 módulo de interface plug-in
- CP 441-2: 2 módulos de interface plug-in

### LEDs:

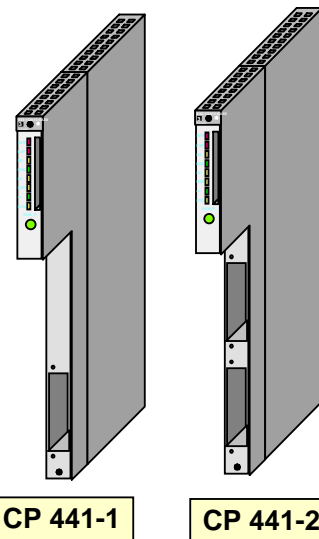
- Envio, Recebimento e Erro

### Baud rate:

- CP 441-1: máx. 38.4 kbaud
- CP 441-2: máx. 76.8 kbaud

### Protocolos:

- Protocolos padrões integrados
- Protocolos não Siemens carregáveis (drivers especiais) - para CP 441-2



- CP441-1: Baixo custo com funcionalidade padrão
- CP441-2: Alta performance para tarefas solicitadas

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.30



Conhecimento em Automação  
Training Center

### Descrição

Alternativa de conexão ponto a ponto com custo compatível e alta performance para barramentos de comunicação. É possível conectar simples equipamentos como leitoras de códigos de barra e impressoras, bem como PLCs.

### Protocolos CP441-1

- Com uma interface plug-in
- 3964 (R) parametrizável
  - Protocolo ASCII parametrizável
  - Impressora

### Protocolos CP441-2

- Com duas interfaces plug-in
- 3964 (R) parametrizável
  - RK 512
  - Protocolo ASCII parametrizável
  - Impressora
  - Protocolos externos carregáveis como:
    - Modbus Mestre / escravo (Modicon)
    - Allen Bradley (protocolo DF1)

### Módulos de Interface

- TTY, conector 9 pinos, módulo de interface ativo/passivo
- V.24 (RS232 C), conector 9 pinos
- RS 422/485, conector 15 pinos

## CP 443-5: Conexão para PROFIBUS

**Formato:** largura simples

**Protocolos:**

- SEND/RCV
- S7 Functions
- FMS (somente CP 443-5 Basic)
- DP Mestre (somente CP 443-5 Extended)

**Baud rate:**

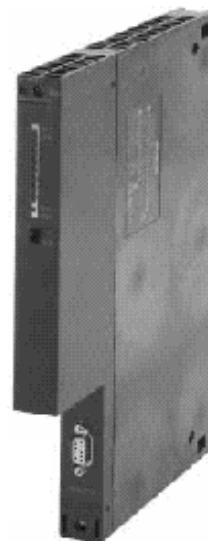
- 9.6 Kbps à 12 Mbps

**Conexão:**

- Cabo elétrico: conector DB9
- Cabo FO: terminal de barramento

**Configuração:**

- NCM S7 para PROFIBUS incluindo FCs e FBs



**CP 443-5 Basic  
CP 443-5 Extended**

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.31



Conhecimento em Automação  
Training Center

#### Descrição

O processador de comunicação CP 443-5 permite conexão do S7-400 à rede PROFIBUS.

#### Protocols

Os seguintes protocolos estão disponíveis:

##### **S7 Functions:**

Para comunicações baseadas na camada (layer) 7 do modelo ISO/OSI entre SIMATIC S7/M7/C7 e PCs.

O protocolo S7 fornece um SFB interface para as CPUs S7 para comunicação dentro da família SIMATIC S7. Adicionalmente, funções para programação, teste, administração de objetos e diagnóstico são fornecidas.

##### **SEND/RCV:**

Para comunicações baseadas na camada (layer) 2 (camada FDL) entre SIMATIC S7, SIMATIC S5, PC/PGs e equipamentos não Siemens. A interface SEND/RCV fornece uma forma simples de realização de comunicação com blocos de dados não estruturados.

A interface SEND/RCV é implementada com:

- Blocos de manipulação no SIMATIC S5
- Chamadas de funções no SIMATIC S7
- Chamadas de funções nos PGs/PCs

##### **PROFIBUS FMS (Fieldbus Message Specification)**

Para comunicação aberta entre SIMATIC S5, S7, PCs/PGs, equipamentos de campo e produtos não Siemens. (EN 50170, Vol. 2, PROFIBUS).

O protocolo FMS permite comunicação orientada a objeto na camada (layer) 7 do modelo ISO/OSI. O típico tamanho de pacote é 240 bytes.

##### **PROFIBUS DP (Periferia Distribuída)**

Para comunicação aberta entre SIMATIC S5, S7, PCs/PGs ou sistemas e equipamentos de campo não Siemens. (EN 50170, Vol. 2, PROFIBUS).

O protocolo DP é projetado para trocas rápidas de pequenas quantidades de dados entre PLCs ou PCs e equipamentos de campo com tempos de resposta < 10 ms.

## IM 467: Interface Mestre PROFIBUS-DP

**Formato:** largura simples

**Protocolos:**

- DP Mestre
- S7 Functions

**Baud rate:**

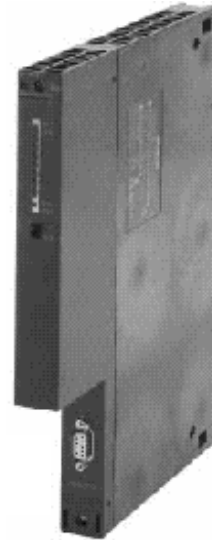
- 9.6 Kbps à 12 Mbps

**Conexão:**

- Cabo elétrico: conector DB9

**Configuração:**

- Configuração e programação possíveis através de PROFIBUS DP



**IM 467**

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.32



Conhecimento em Automação  
Training Center

### Descrição

O módulo de interface IM 467 é planejado para operação em um sistema PLC S7-400. Este possibilita a conexão do S7-400 ao PROFIBUS DP.

### Protocolos

O IM 467 oferece dois serviços de comunicação:

#### PROFIBUS DP

O IM 467 é um Mestre PROFIBUS DP de acordo com EN 50 170. A configuração ocorre totalmente com STEP 7. A performance é idêntica, em princípio, as interfaces integrada PROFIBUS DP nos módulos CPU.

Nenhuma chamada de função em STEP 7 no programa do usuário é necessária para comunicação DP.

#### S7 Functions

As funções S7 garantem comunicação ótima e simples em uma solução de automação SIMATIC S7/M7/C7. As seguintes funções S7 são habilitadas para o IM 467:

- Funções PG através de PROFIBUS DP
- Funções HMI através de PROFIBUS DP

A comunicação ocorre sem configurações adicionais no IM 467.

As funções S7 podem ser utilizadas sozinhas ou em paralelo ao protocolo PROFIBUS DP. Se elas utilizadas em paralelo a comunicação DP, então esta apresenta repercussão no ciclo de tempo do barramento PROFIBUS DP.

## CP 443-1: Conexão para Ethernet Industrial

**Formato:** largura simples

**Protocolos:**

- SEND/RCV e S7 Functions na Pilha de Transporte ISO (CP 443-1) e Pilha TCP/IP (CP 443-1 TCP/IP)
- CP444: MMS/MAP

**Conexões:**

- S7 Functions: máx. 48 conexões
- SEND/RCV: máx. 64 conexões

**Funções:**

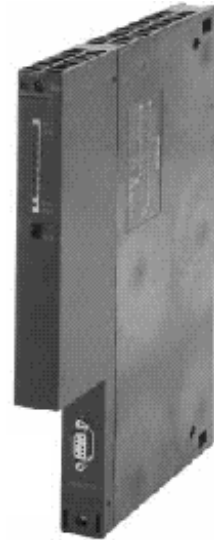
- Capacidade de protocolos múltiplos
- Programação remota através de LAN e WAN (somente CP443-1 TCP/IP)

**Conexão:**

- Chaveamento automático entre AUI e conexão par trançado

**Configuração:**

- NCM-S7 para Ethernet Industrial incluindo chamadas de funções para SEND/RCV



**CP 443-1  
CP 443-1 TCP/IP  
CP 444**

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.33



Conhecimento em Automação  
Training Center

### Descrição

Os processadores de comunicação CP 443-1 e CP 443-1 TCP/IP permitem conexão do S7-400 à rede Ethernet Industrial.

### Protocolos

Os seguintes protocolos estão disponíveis:

#### **S7 Functions**

Para comunicações baseadas na camada (layer) 7 do modelo ISO/OSI entre SIMATIC S7/M7/C7 e PCs.

#### **SEND/RCV**

Para comunicações baseadas na camada (layer) 4 (Pilha de Transporte ISO no CP 443-1 e Pilha de Transporte TCP no CP 443-1 TCP/IP) entre SIMATIC S7, SIMATIC S5 e PC/PGs.

#### **MMS/MAP**

Para comunicações abertas baseadas na camada (layer) 7 entre SIMATIC S7, SIMATIC S5, PCs/PGs e sistemas não Siemens.

Esta funcionalidade é fornecida pelo CP 444.



## CP 443-1 IT: Conexão para Internet

### Mesmo formato e funcionalidade do CP 443-1 TCP:

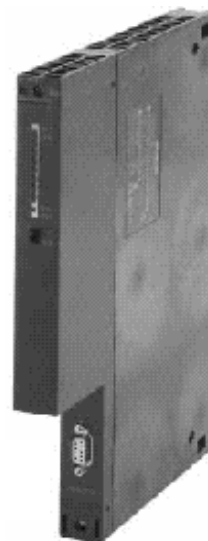
- S7 Functions
- Send/Receive através de RFC 1006 e UDP

### Funcionalidade adicional Internet: CP 443-1 IT é um WWW server

- Páginas HTML e applets on board para S7 Functions
- WWW server utilizados para operação de controle/monitoração de pequenos controladores
- Nenhum custo no cliente final
- Plataforma independente
- Filosofia de operação de Internet familiar

### E-mail client:

- Uso de e-mail para alertas de falhas
- Acesso a telefones celulares, pagers, PC, Fax, etc.
- Uma chamada, diferentes sistemas de acessos simultâneos, isto é, nenhum software a parte para cada sistema



CP 443-1 IT

### SIMATIC S7

Siemens AG 1999. todas rights reserved.

Date: 04.10.2007  
File: PRO2\_11P.34



Conhecimento em Automação  
Training Center

### Descrição

O processador de comunicação CP 443-1 IT permite você conectar o S7-400 à Internet.

### Protocolos

Os seguintes protocolos estão disponíveis para o CP 443-1 IT:

#### S7 Functions

Para comunicação baseada na camada (layer) 7 do modelo ISO/OSI entre SIMATIC S7/M7/C7 e PCs.

#### SEND/RCV

Para comunicação baseada na camada (layer) 4 (Pilha de Transporte TCP) entre SIMATIC S7, SIMATIC S5 e PC/PGs.

### Comunicação através da Internet

Da mesma forma que as facilidades de comunicação industrial, esta CP Internet fornece acesso a Internet. Isto significa:

- O usuário pode se logar no sistema de qualquer lugar usando um password.
- Você pode ler dados de processo e operação de uma planta com qualquer browser de Internet (Internet Explorer, Netscape, etc.). (páginas Integraís HTML para informações do sistema).

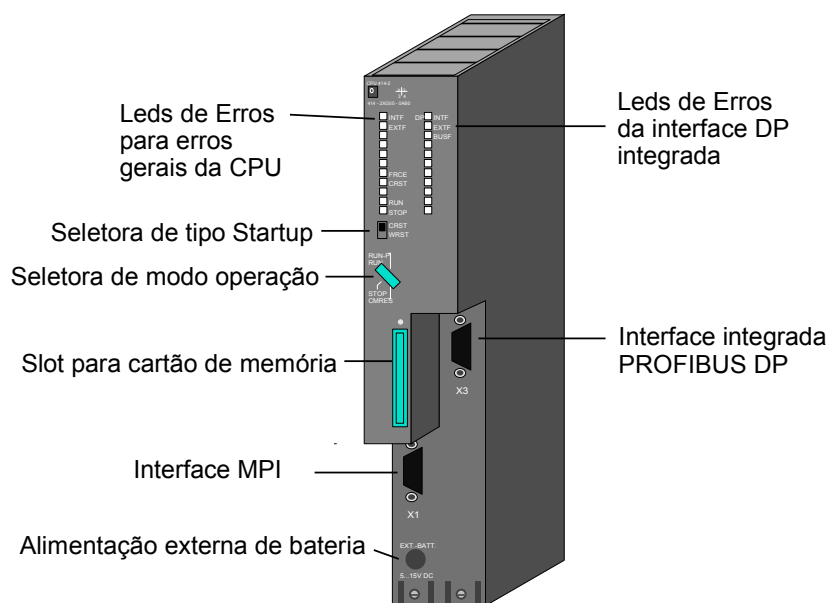
Intervenção do operador também é possível, permitindo serviços a serem realizados de qualquer lugar do mundo.

O CP Internet também habilita você a enviar todas as informações importantes de produção ou estados da planta de qualquer lugar do mundo, a saber:

- por e-mail através da Internet
- para telefones celulares ou máquinas de fax
- to external PCs
- para pagers ou palmtops com acesso a Internet.



## I/O Distribuído e Atribuição de Parâmetros



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.1



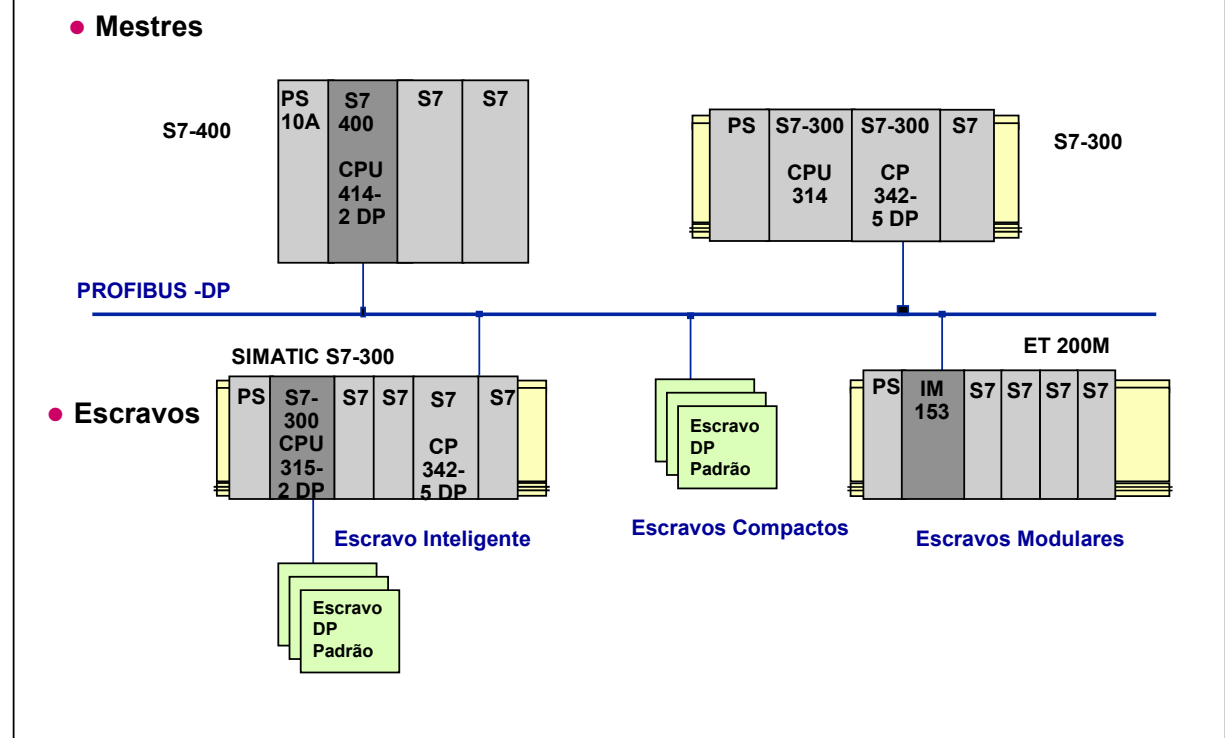
Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Estrutura de um Sistema PROFIBUS DP .....	2
Métodos de Comunicação PROFIBUS .....	3
Ciclo de Tempo de um Sistema Mono Mestre PROFIBUS DP .....	4
PROFIBUS Mestre no SIMATIC S7 .....	5
Escravos DP disponíveis .....	6
Resistor de Terminação PROFIBUS DP .....	7
Configurando um Sistema DP Mestre .....	8
Configurando Escravos DP Compactos e Modulares .....	9
Configurando Escravos DP Inteligentes em um Sistema DP Mestre (p.ex. CPU 315-2) .....	10
Inserindo Escravos DP Inteligentes em um Sistema Mestre .....	11
Análise de Erros/Falhas no OB 86 quando ocorre falhas em Escravos .....	12
Diagnose de Escravos com SFC 13 (DPNRM_DG) .....	13
Lendo Consistência de Dados dos Escravos DP Padrões com SFC 14 .....	14
Escrevendo Consistência de Dados dos Escravos DP Padrões com SFC 15 .....	15
Sincronizando Escravos DP com SFC 11 (DPSYC_FR) .....	16
Instalação posterior de Escravos PROFIBUS DP .....	17

## Estrutura de um Sistema PROFIBUS DP



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.2



Conhecimento em Automação  
Training Center

### Vista Geral

Equipamentos instalados no campo para automação de processos, como sensores, atuadores, transdutores e acionamentos estão sendo fabricados cada vez mais utilizando sistemas de comunicação de campo para troca de informações com unidades de controle de alto nível.

O PROFIBUS é um sistema de comunicação de campo que pode ser usado por todos os equipamentos de automação, tais como PLCs, PCs, interfaces homem máquina, atuadores e sensores, para troca de dados.

### PROFIBUS DP

PROFIBUS DP é um protocolo otimizado para velocidade, o qual foi especialmente projetado para comunicação entre PLCs (Mestres DP) e I/Os distribuídos (Escravos DP).

PROFIBUS DP é de baixo custo e flexível para substituir a transmissão dos incômodos sinais paralelos 24V e linhas 20mA.

PROFIBUS DP é baseado na DIN 19245 Parte 1 e extensões especificados por usuários da DIN 19245 Part 3. No curso do processo de padronização de comunicações de campo europeu, o PROFIBUS DP foi integrada nos padrões de comunicações de campo europeu EN 50170.

### Mestres

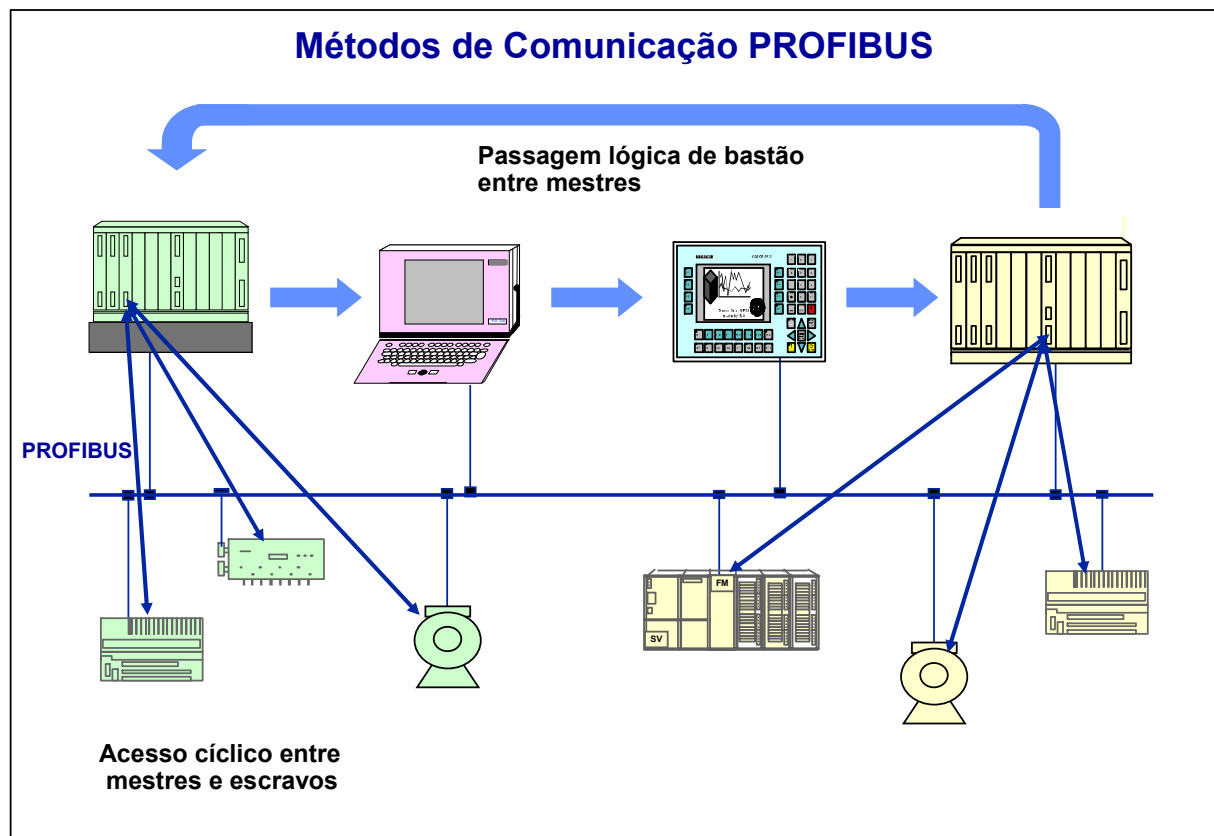
PROFIBUS faz distinção entre mestres e escravos.

Os mestres PROFIBUS são mandatários no tráfego de dados na rede. Um mestre pode enviar mensagens sem receber requisição para isto, fornecendo a posse do bastão para poder acessar o barramento de comunicação.

O mestres também são referenciados no protocolo PROFIBUS como nós ativos.

### Escravos

Os escravos PROFIBUS são simples equipamentos de I/O, tais como atuadores, sensores, transdutores, etc. Eles não recebem o bastão, ou seja, eles somente podem reconhecer o recebimento de mensagens (dados) requisitados por um mestre. Escravos são nós passivos.

**SIMATIC S7**

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.3Conhecimento em Automação  
Training Center**Controle de Acesso ao Barramento**

O método de controle de acesso ao barramento determina quando um nó pode enviar dados. Isto é essencial que somente um nó tenha o direito de envio de dados a cada intervalo de tempo.

O protocolo PROFIBUS fornece para duas requisições básicas atreladas no barramento um método de controle de acesso:

- Para comunicação entre estações complexas de mesmo nível (mestres), elas devem se assegurar que cada uma destas estações tenha oportunidade suficiente de operar com suas tarefas de comunicação nos intervalos definidos.
- Para comunicação entre um mestre complexo e simples I/Os associados a ele (escravos), uma cíclica troca de dados em tempo real deve ser implementada com uma pequena sobra sempre que possível.

O método de controle de acesso ao barramento PROFIBUS portanto emprega passagem de bastão para comunicação entre mestres complexos e princípio mestre-escravo para comunicação entre mestres e simples equipamentos de I/O (escravos).

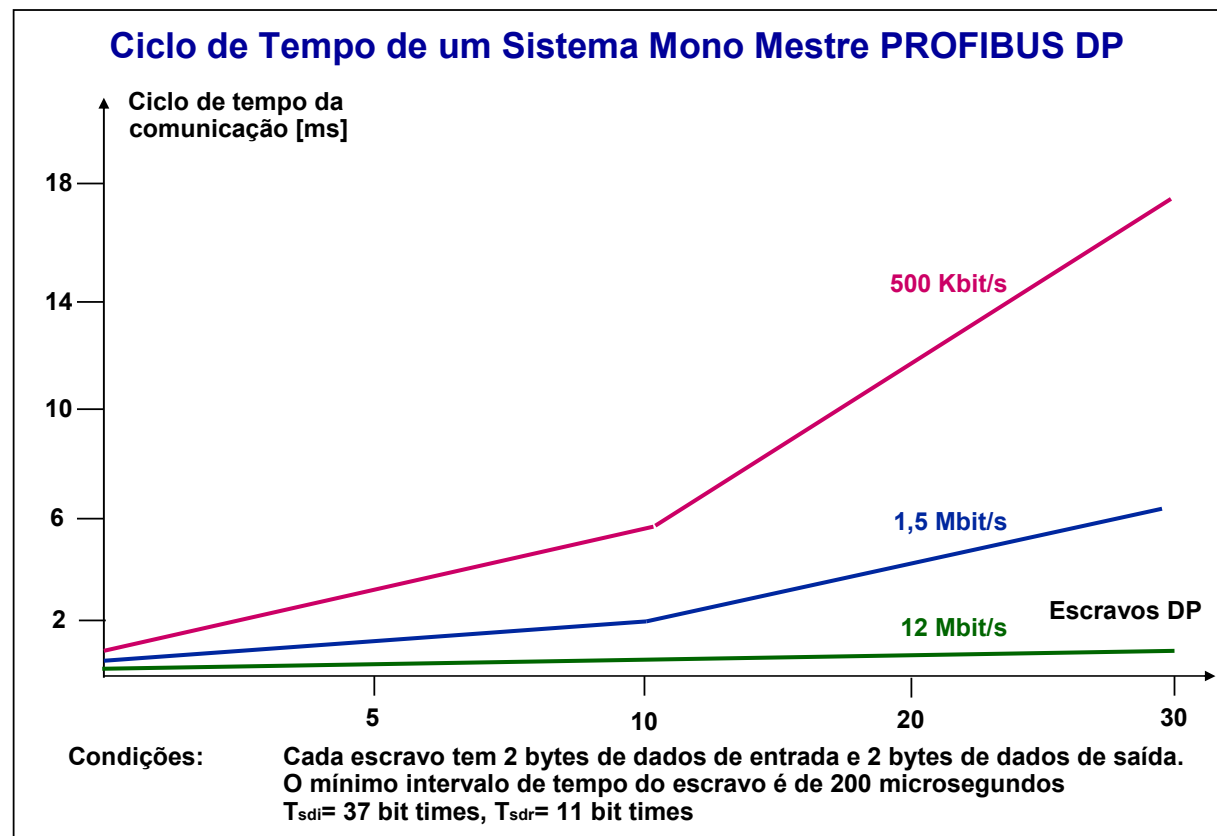
**Método de Passagem de Bastão**

O método de passagem de bastão assegura que o acesso direto ao barramento (bastão) se dará no exato instante definido.

O bastão, uma moldura de mensagem especial que passa o direito de enviar de um mestre para o próximo, deve ser dado uma vez para cada mestre por volta dentro de um tempo máximo de circulação do bastão.

**Princípio Mestre-escravo**

O princípio mestre-escravo habilita o mestre (nó ativo) que esteja de posse do bastão endereçar os escravos parametrizados a ele (nós passivos). O mestre pode enviar mensagens (dados do usuário) para outros escravos ou buscar mensagens (dados do usuário) dos escravos.



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.4Conhecimento em Automação  
Training Center**PROFIBUS DP**

O protocolo PROFIBUS DP é projetado para troca rápida de dados no nível sensor / atuador. Neste nível as unidades de controle central, tais como PLCs, se comunicam com equipamentos de entradas e saídas distribuídas através de uma conexão serial de alta velocidade. A troca de dados com estes equipamentos distribuídos é preponderantemente cíclica.

O controlador central (mestre) lê os dados de entrada dos escravos e escreve as informações de saída nos escravos. O ciclo de tempo da comunicação deve ser menor do que o ciclo da varredura do PLC.

**Nota**

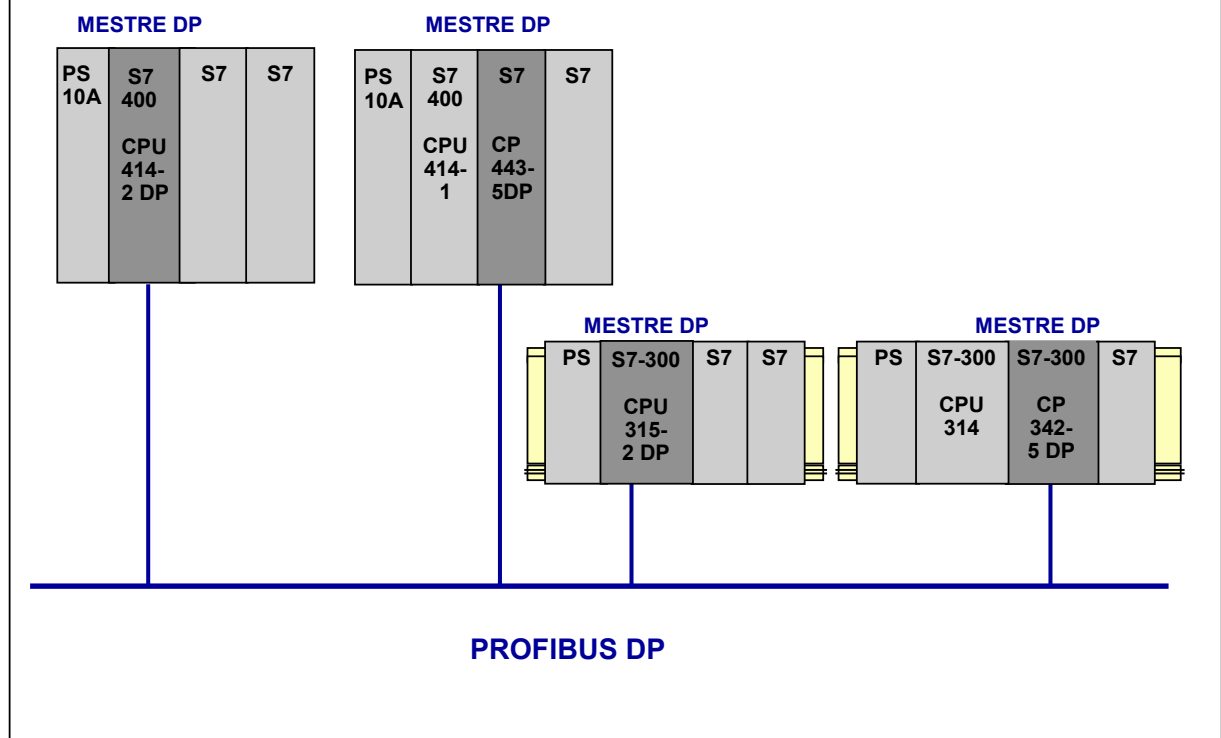
O protocolo PROFIBUS DP não pode ser utilizado para troca de informações entre mestres.

**Velocidade**

Para a transmissão de 512 bits de dados de entrada e 512 bits de dados de saída divididos entre 32 nós PROFIBUS DP leva aproximadamente 6 ms com uma velocidade de transmissão de 1.5 Mbit/s e menos do que 2 ms em 12 Mbit/s.

A velocidade superior deste protocolo supera a do protocolo PROFIBUS FMS principalmente devido ao fato de que os dados de entrada e saída são transferidos em um ciclo de mensagem usando a camada (layer) 2 de serviço de envio e recebimento de dados.

## PROFIBUS Mestre no SIMATIC S7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.5



Conhecimento em Automação  
Training Center

### Vista Geral

O SIMATIC S7/M7 mantém a tendência em direção à automação distribuída pela integração de I/O distribuídos nos sistemas de automação. A tecnologia de automação inovadora do SIMATIC S7/M7 forma uma parceria ideal com a comunicação de campo internacionalmente estabelecida PROFIBUS DP/PA e estações de I/O distribuídos.

### Mestre PROFIBUS

Os PLCs S7-300 e S7-400 podem se conectar ao PROFIBUS como mestres individualmente através de CPUs com interface integrada PROFIBUS DP ou através de processadores de comunicação (CPs).

As CPUs com interface integrada PROFIBUS DP permitem a você configurar sistemas de automação distribuída com velocidades de comunicação de até 12 Mbaud.

### Integração

A total integração de sistemas de PLC e I/Os distribuídos tem as seguintes vantagens para o usuário:

- Configuração uniforme: Você configura o central e os I/Os distribuídos com STEP 7. Isto representa uma ferramenta de configuração uniforme para o usuário, independentemente do tipo de solução de automação.
- Programação centralizada e distribuída: você programa o PLC com STEP 7 independente do tipo de configuração. Isto significa que você pode escrever programas sem se preocupar com a configuração final do hardware.
- Performance de sistema total, se em uma configuração centralizada ou distribuída: SIMATIC S7/M7 oferece poderoso suporte de sistema. Isto inclui softwares de parametrização de I/Os, uma extensa gama de facilidades de diagnósticos e módulos com funções fáceis de conectar.
- Programação, teste e startup via PROFIBUS-DP: Chamada de estruturas de automação distribuída para facilitar o startup. Com STEP 7 você pode programar, testar e colocar em operação (start up) o PLC central de um ponto do campo com a mesma facilidade que você tem da porta de comunicação na CPU com o equipamento de programação.

## Escravos DP disponíveis



ET 200M



ET 200U

Escravos modulares consistem de um módulo de Interface e módulos da família S7-300 (ET 200M) ou família S5 (ET 200U).



ET 200B



ET 200L

Pequenos, estações compactas de I/Os (grau de Proteção IP 20) com canais de entrada e saída Integrados.



ET 200X



ET 200S

Módulos de Interface com módulos de entrada/saída, chaves de partidas, etc.  
Grau de proteção: ET 200X: IP 65/67, ET 200S: IP 20



CPU 215



CPU 315-2 DP

Escravos DP inteligentes das famílias S7-200 e S7-300 para pré processamento de dados.



CPU 316-2 DP



CPU 318-2 DP



CP 342-5

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.6



Conhecimento em Automação  
Training Center

#### Escravos Modulares ET 200M

O ET200M consiste de um módulo de interface IM153-1 que é conectado a um mestre S7/M7-PROFIBUS. Todos os módulos S7-300 endereçados através do barramento P pode ser inserido no ET 200M.

Máximo espaço de endereço por ET 200M: 128/128 bytes cada para entradas/saídas com um máximo de 12 Mbaud.

#### Escravos Compactos ET 200L e ET 200B

Ambos ET 200L e ET 200B consistem de um bloco terminal e um bloco eletrônico. Existem blocos eletrônicos com canais digitais e analógicos. O ET 200L é usado onde poucas entradas e saídas são necessárias com baud rates de até 1.5 Mbaud.

O ET 200B é usado onde existe um número limitado de espaço de montagem. O baud rate máximo é de 12Mbaud.

#### Escravos Compactos ET 200C

O compacto ET 200C com o elevado grau de proteção IP66/IP77 é projetado para aplicação em ambientes industriais agressivos. (Também pode ser usado ao tempo). Com um baud rate máximo de até 12Mbaud para entradas/saídas digital e até 1.5 Mbaud para entradas/saídas analógicas.

#### Escravos Modulares ET 200X

O ET 200X é uma estação I/O compacta com o elevado grau de proteção IP 65/IP 67 e consiste de um módulo básico e módulos de expansão (p.ex. módulos de entrada/saída, mestre AS-interface, módulos chaves de partidas, módulos pneumáticos, fontes de alimentação SITOP).

#### Escravos Modulares ET 200S

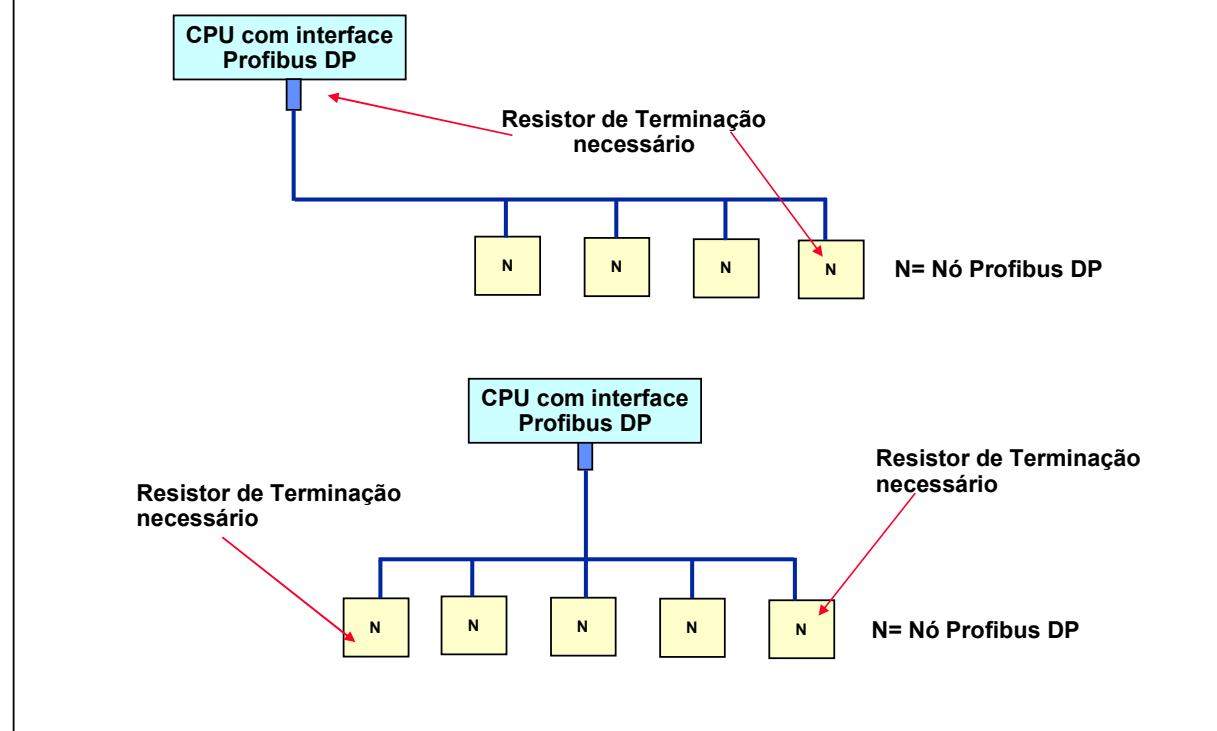
O ET 200S é uma estação I/O distribuída com grau de proteção IP 20. Seu elevado projeto modular habilita-o a adaptar-se rápida e perfeitamente em qualquer aplicação.

Os ET 200S consistem de módulos de interface PROFIBUS DP, módulos eletrônicos digitais e analógicos, módulos de função tecnológico (p.ex. contador, controle de posicionamento) e chaves de partida.

#### Escravos Inteligentes

P.ex. CPU 315-2, CP 342-5 ou S5-95-PROFIBUS com funcionalidade escrava.

## Resistor de Terminação PROFIBUS DP



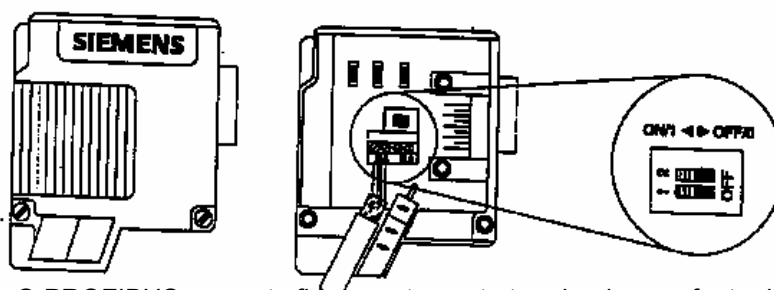
### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.7Conhecimento em Automação  
Training Center

### Ajustes do Resistor de Terminação

Manter o resistor de terminação do primeiro e do último conectores da rede na posição ON. Para fazer isto, você abre a cobertura do conector de barramento e ajuste a chave na posição ON (ver diagrama).



O PROFIBUS somente fica corretamente terminado se a fonte de alimentação do nó no qual o resistor de terminação é inserido está atualmente chaveado em ON. Se não for este o caso, o PROFIBUS também pode ser terminado com um resistor de terminação RS485 ativo (6ES7972-0DA00-0AA0). O resistor de terminação então recebe uma tensão de alimentação permanente separada daquela de outros componentes I/O ou alimentado por este antes dos I/Os.

A terminação do sistema de barramento habilita os nós (p.ex. ET 200L) a se conectados e desconectados quando necessário, sem causar mau funcionamento.

### Comprimento dos Cabos

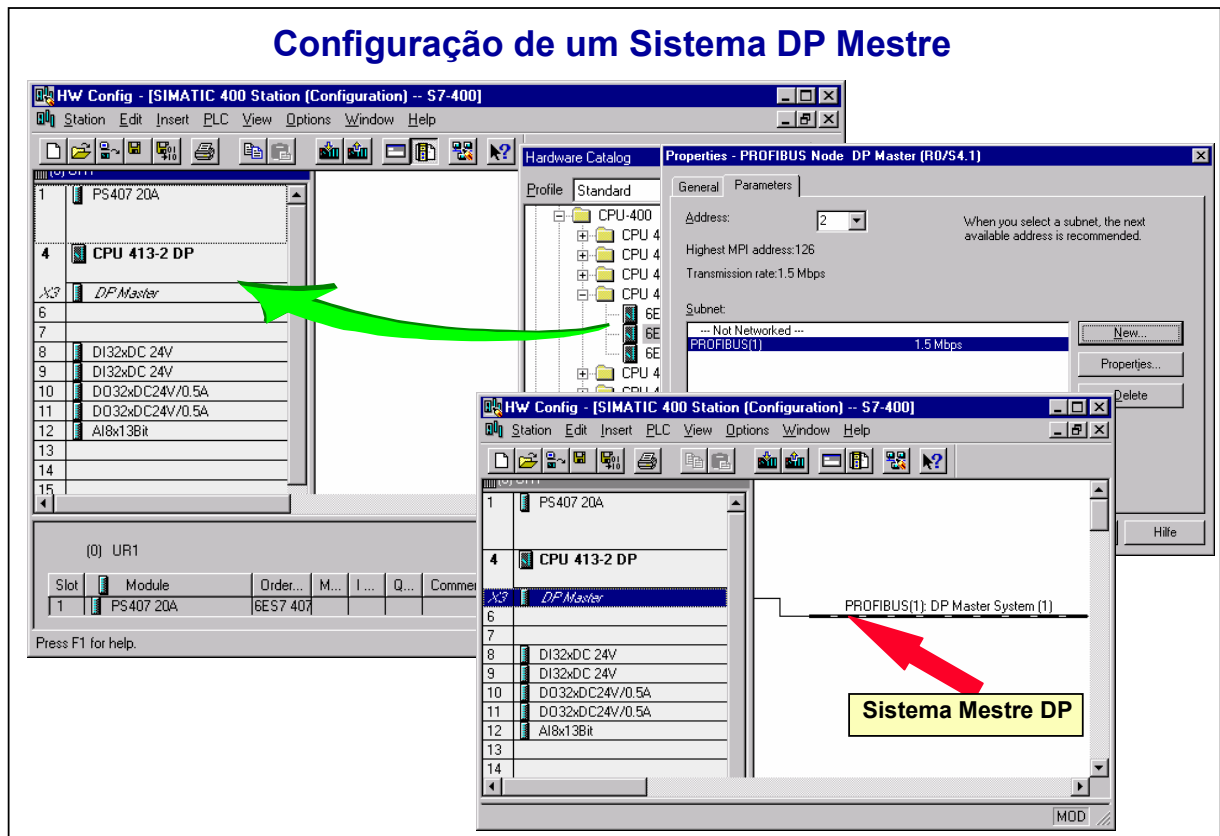
O comprimento máximo de um segmento Profibus depende do baud rate:

Baud rate	Comprimento do Segmento
9.6 to 187.5 Kbaud	10000 m
500 Kbaud	400 m
1.5 Mbaud	200 m
3 to 12 Mbaud	100 m

O máximo comprimento de um segmento para MPI é 50 m. Até 9 repetidores (repeaters) podem ser conectados em uma fila.



## Configuração de um Sistema DP Mestre



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.8



Conhecimento em Automação  
Training Center

### I/O Distribuído

Todos os sistemas mestres são constituídos de um mestre DP master e escravos DP que são conectados através de um cabo barramento e que se comunicam através de protocolo PROFIBUS DP são designados como I/Os distribuídos.

### Mestre DP

Como mestre DP você pode instalar:

- CPU S7 com interface mestre DP integrada (p.ex. CPU 414-2, etc.);
- Submódulo de interface, que é atribuída a uma CPU M7 / FM M7;
- CP em conexão com uma CPU (p.ex. CP 443-5, etc.)

### Ajustando um Mestre DP

Para configurar um sistema mestre, proceder como a seguir:

1. Selecionar um mestre DP da janela "Hardware Catalog".
2. Usando marcar e arrastar, insira o módulo na linha permitida do bastidor.

A caixa de diálogo "Properties - PROFIBUS Nodes" é aberta. Neste diálogo você pode estabelecer as seguintes propriedades:

- configurar uma nova subrede PROFIBUS ou selecionar uma existente.
- ajustar as propriedades da subrede PROFIBUS (baud rate, etc.).
- estabelecer o endereço PROFIBUS do mestre DP.

3. Reconhecer a configuração com "O.K.". Os seguintes símbolos aparecem:

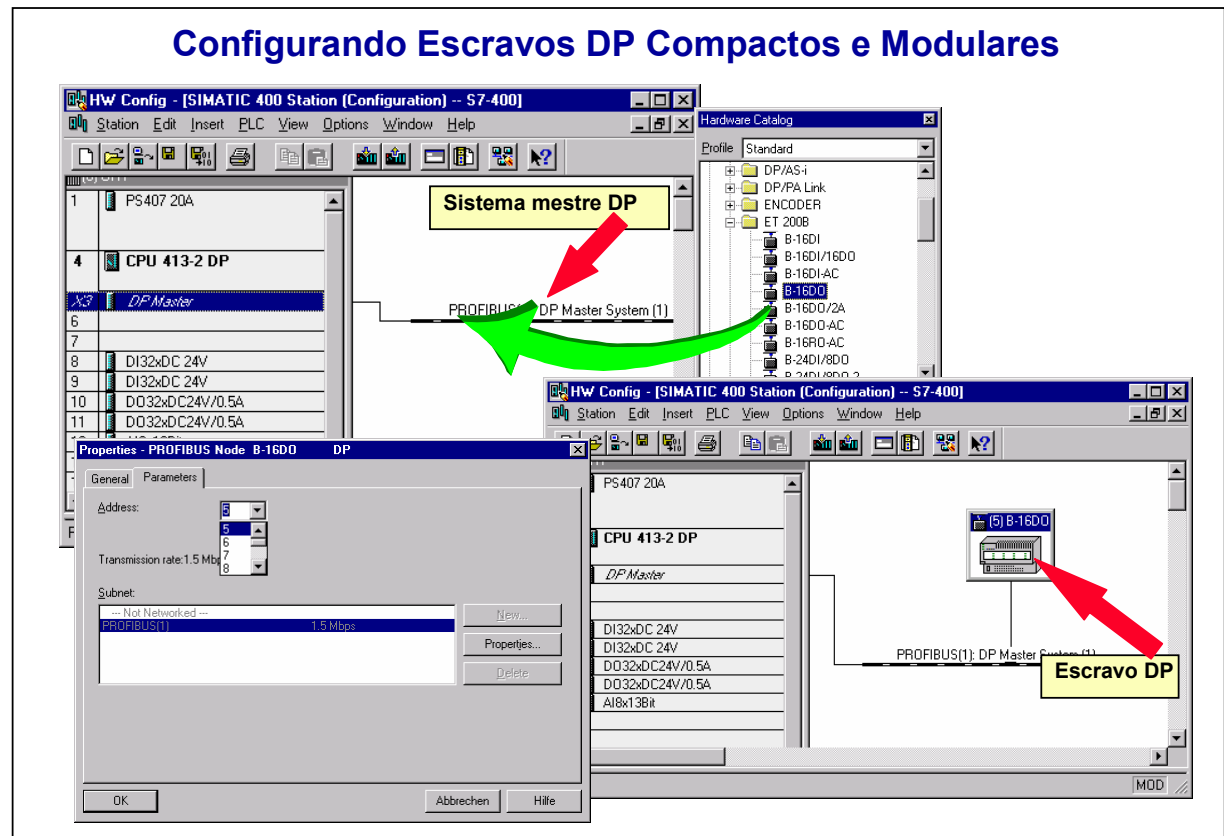
 para sistema mestre DP. Este símbolo é usado como "hanger" (pindurador) para os escravos DP.

### Nota

Você pode ter operação mono-mestre bem como operação multi-mestre na subrede PROFIBUS DP. Em operação mono-mestre somente um mestre DP é operado na subrede PROFIBUS, em operação multi-mestre diversos mestres DP com seus respectivos sistemas mestres são operados em uma subrede PROFIBUS.



## Configurando Escravos DP Compactos e Modulares



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.9




Conhecimento em Automação  
Training Center

### Escravos DP

- Módulos com entradas e saídas digitais/analógicas integradas (escravo DP compacto, p.ex. ET200B).
- Módulos de interface com módulos S5 ou S7 (escravos modulares DP, p.ex. ET200M).
- Estações S7-200/300 com módulos que suportam funções "Escravo Inteligente" (p.ex. CPU 215-DP, CPU 315-2).

### Selecionando Escravos DP

De forma a configurar um escravo DP, proceder como a seguir:

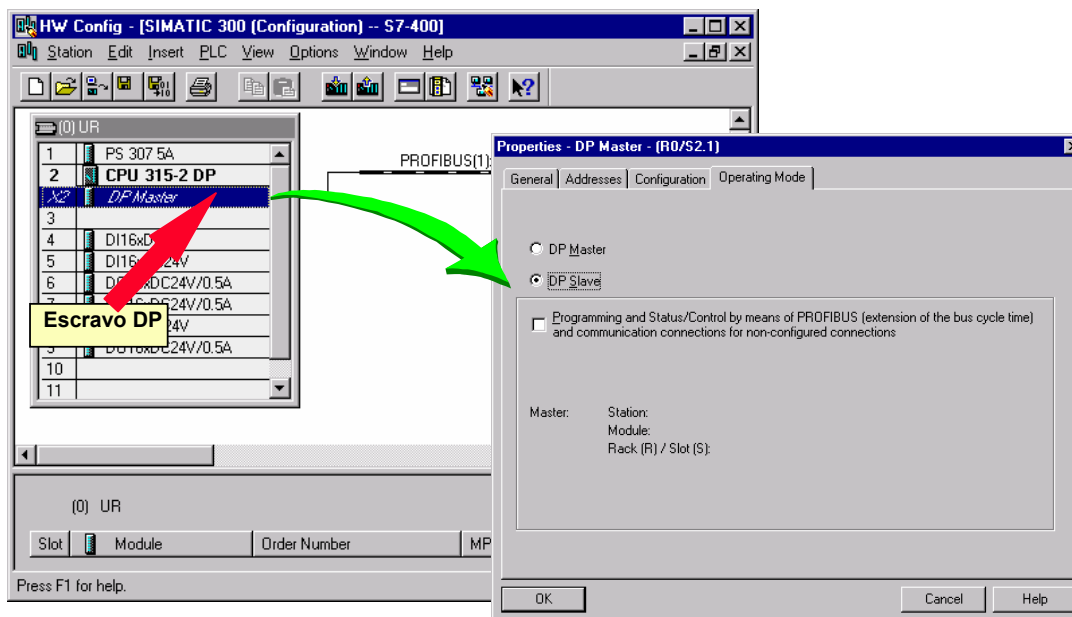
1. Selecione o escravo DP compacto desejado (p.ex. ET200B) ou o módulo de interface (p.ex. IM153 para ET200M) para um escravo modular do "Hardware Catalog".
2. Arraste o símbolo para dentro do símbolo de sistema mas: 

A caixa de diálogo "Properties - PROFIBUS Nodes" é aberta. Aqui você pode ajustar:

  - propriedades da subrede PROFIBUS (baud rate, etc.).
  - o endereço PROFIBUS do escravo DP.
3. Reconheça as configurações com "O.K.". Uma tabela de configuração é anexada ao símbolo, que representa o complemento de I/O do escravo compacto ou o bastidor do escravo modular.
4. Para um escravo DP modular, você agora insere módulos desejados do "Hardware Catalog" na tabela de configuração.

O endereçamento e atribuição de parâmetros dos módulos então ocorre do mesmo modo que na configuração central.

## Configurando Escravos DP Inteligentes em um Sistema DP Mestre (p.ex. CPU 315-2)



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.10



Conhecimento em Automação  
Training Center

### Escravos Inteligentes

A característica fundamental de um escravo DP inteligente é que a entrada/saída de dados não estão diretamente disponíveis ao mestre DP de uma entrada/saída real, mas pré processadas pela CPU.

Com um escravo DP inteligente, o mestre DP não acessa as entradas/saídas do escravo DP inteligente, mas a área de endereços da CPU "pré processadora". O programa do usuário da CPU pré processadora deve tomar cuidado da troca de dados entre a área de endereços e a área de entrada/saída.

### Nota

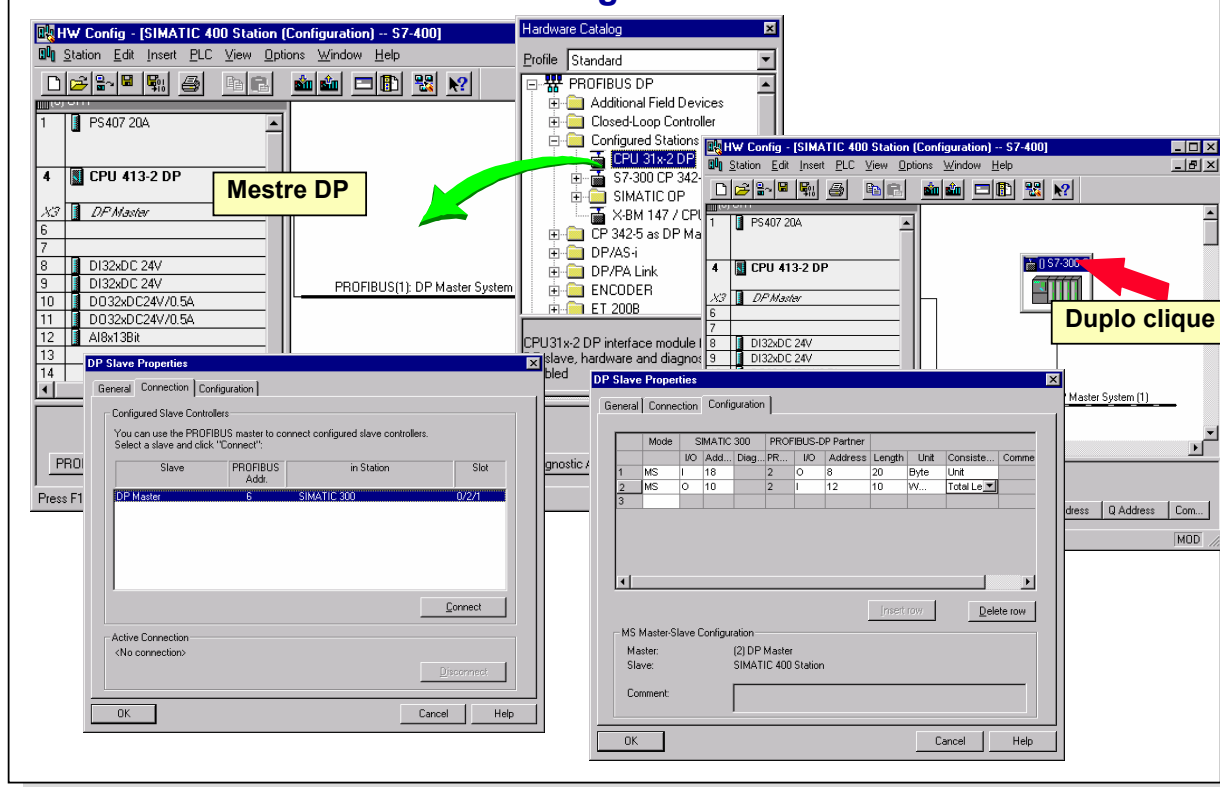
Um escravo DP inteligente (p.ex. CPU 315-2 DP) não pode ser configurada simultaneamente como um mestre DP e um escravo DP. Uma CPU 315-2 DP configurada como um escravo DP não pode simultaneamente ser mestre DP para outras estações escravas DP.

### Configurando Escravos DP

De forma a configurar uma CPU 315-2 DP como um escravo DP inteligente, proceda como a seguir:

1. Insira uma estação S7-300 em seu projeto.
2. Abra o editor HW Config, pela seleção da estação então com duplo clique no símbolo "Hardware".
3. Insira uma CPU 315-2 DP do catálogo de Hardware para dentro da tabela de configuração.
4. Duplo clique na linha 2.1 da tabela de configuração. O diálogo "Properties - DP Master" é aberto.
5. Ative a opção "Use Controller as Slave" na página da tabela "Slave Configuration".
6. Especifique outros parâmetros PROFIBUS da CPU 315-2-DP.
7. Confirme os ajustes com "O.K.".

## Inserindo Escravos DP Inteligentes em um Sistema Mestre



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.11



Conhecimento em Automação  
Training Center

### Inserindo um Escravo DP Inteligente

De forma a inserir uma CPU 315-2 DP como um escravo DP inteligente em um sistema mestre, proceda como a seguir:

1. No seu projeto insira uma estação com capacidade mestre DP (p.ex. S7-400).
2. Abra o editor HW Config, pela seleção da estação e então um duplo clique no símbolo "Hardware".
3. Inserir um mestre DP (p.ex. CPU 414-2 DP) do "Hardware Catalog" na tabela de configuração.
4. Duplo clique na linha "DP Master" da tabela de configuração. O diálogo "Properties - DP Master" é aberto.
5. Especifique todos os parâmetros mestre DP PROFIBUS e salve a configuração com o botão OK.
6. Usando marca e arrasta, arraste a CPU 315-2 DP do "Hardware Catalog" (que já contém estações configuradas) para o sistema mestre.
7. Duplo clique na segunda linha do escravo DP (o nome do escravo DP, p.ex. CPU 315-2 DP é encontrado lá) e selecione a tabela "Connection". Uma lista de todas configuradas, escravos DP inteligentes é mostrado nesta página da tabela.
8. Selecione o escravo DP inteligente desejado e clique o botão de comando "Connect".
9. Selecione a tabela "Slave Configuration" e atribua os endereços mestre e escravo de um com o outro.  
Áreas de entrada do mestre DP são áreas de saída do escravo DP e vice versa.
10. Confirme a configuração com "O.K.". Os dados pré processados pela CPU 315-2 DP são agora atribuídos a CPU 414-2 DP que é um sistema mestre com um escravo inteligente.

### Nota

Para uma correta inicialização após estabelecida a alimentação do sistema mestre DP e escravo DP inteligente, os Obs de erro associados (OB 85, OB 86, etc.) devem ser transferidos para as respectivas CPUs.

## Análise Erros/Falhas no OB 86 quando ocorre falhas em Escravos

Address	Decl.	Name	Type	Initial	Comment
0.0	temp	OB86_EV_CLASS	BYTE		16#38/39 Event clas
1.0	temp	OB86_FLT_ID	BYTE		16#C1/C4/C5, Fault
2.0	temp	OB86_PRIORITY	BYTE		26/28 (Priority of
3.0	temp	OB86_OB_NUMBER	BYTE		86 (Organization b.
4.0	temp	OB86_RESERVED_1	BYTE		Reserved for system
5.0	temp	OB86_RESERVED_2	BYTE		Reserved for system
6.0	temp	OB86_MDL_ADDR	WORD		Depending on fault
8.0	temp	OB86_RACKS_FLTD	ARRAY[0..31]		
*0.1	temp		BOOL		
12.0	temp	OB86 DATE TIME	DATE AND TIME		Date and time OB86

OB86 : "Loss Of Rack Fault"

**Network 1:** Title:

```

L   #OB86_FLT_ID           //DP-Station failure
L   B#16#C4
==I
=   M   0.4
  
```

Press F1 for help. Offline Abs Nw 1 Ln 4 INS MOD

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.12Conhecimento em Automação  
Training Center

### Falha da Estação

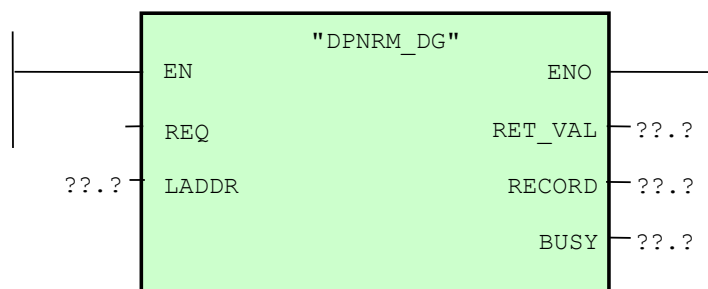
O sistema operacional da CPU (CPU 315-2DP ou S7-400) ativa o OB86, se a falha do bastidor, da subrede ou I/O distribuído é detetado, se o evento é identificado como começando ou terminando.

Se você não programou o OB86 e um erro ocorre a CPU vai para o modo STOP.

### Variáveis no OB86

- OB86\_FLT\_ID: B#16#C4 //falha na estação DP
  - OB86\_FLT\_ID: B#16#C5 //estação DP faltante
  - OB86\_MDL\_ADDR: Endereço base lógico do mestre DP (endereço de diagnóstico)
  - OB86\_RACKS\_FLTD: ==> Muda tipo de dado para DWORD
- Conteúdos:
- Bit 0 a 7: Número da estação DP (endereço PROFIBUS)
  - Bit 8 a 15: ID de subrede DP
  - Bit 16 a 30: Endereço base lógico do escravo DP (endereço de diagnóstico)
  - Bit 31: Identificador I/O

## Diagnose de Escravos com SFC 13 (DPNRM\_DG)



Parâmetro	Declaração	Tipo dado	Área de Memória	Descrição
REQ	INPUT	BOOL	I, Q, M, D, L, Const.	REQ = 1: Requisição para leitura
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Endereço diagnóstico configurado do escravo DP
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Se um erro ocorrer durante processamento da função, o valor retornado contém um código de erro. Se nenhum erro ocorrer, RET_VAL contém o comprimento dos dados transmitidos atualmente.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Área de destino p/leitura de dados de diagnóstico. Somente tipos de dados BYTE são permitidos.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	O mínimo comprimento do arquivo de dados a serem lidos da área de destino é 6. BUSY = 1: A leitura ainda não foi completada.

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.13Conhecimento em Automação  
Training Center

**Diagnose do Escravo** Com SFC 13 "DPNRM\_DG" você lê os dados de diagnóstico de um escravo DP na forma estipulada na EN 50 170.

Se nenhum erro ocorrer durante a transmissão, os dados lidos são inseridos na área de destino especificada pelo RECORD (OUT 2).

Você inicia a função leitura pela atribuição de 1 no parâmetro de entrada REQ (IN0) quando você chama SFC 13.

### Estrutura de Diagnose Escravo

A estrutura básica de dados de diagnóstico do escravo é mostrada na seguinte tabela. Para mais informações favor ver manuais dos escravos DP (por exemplo, números de erro no manual NCM-S7).

#### Estrutura básica de diagnose do escravo

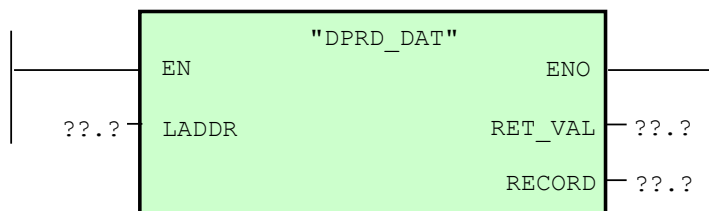
Byte	Significado
0	Estado da estação 1
1	Estado da estação 2
2	Estado da estação 3
3	Número da estação mestre
5	Identificação do fabricante (byte baixo)
6...	Demais diagnoses do escravo específico

### Nota

No caso dos escravos padrões para os quais o número do dado de diagnose padrão é maior que 240 bytes e não mais do que 244 bytes, os primeiros 240 bytes são transferidos para a área de destino e o correspondente bit de estouro (overflow) é setado nos dados.

## Lendo Consistência Dados dos Escravos DP Padrões com SFC 14

- Você necessita da SFC 14 "DPRD\_DAT" para ler mais de quatro bytes consecutivos de dados (dados consistentes).



Parâmetro	Declaração	Tipo Dado	Área Memória	Descrição
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Configura endereço de partida na área de entrada do módulo do qual o dado será lido.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Se um erro ocorrer durante o processamento da função o valor retornado contém um código de erro.
RECORD	OUTPUT	ANY	I, Q, M, D, L	Área de destino para o dado do usuário lido. Ele deve ser exatamente do mesmo tamanho que a área que você configurou para o módulo selecionado com o STEP 7. Somente dados do tipo BYTE são permitidos.

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.14



Conhecimento em Automação  
Training Center

### Função

Com a SFC 14 "DPRD\_DAT" você lê dados consistentes de um escravo DP padrão.

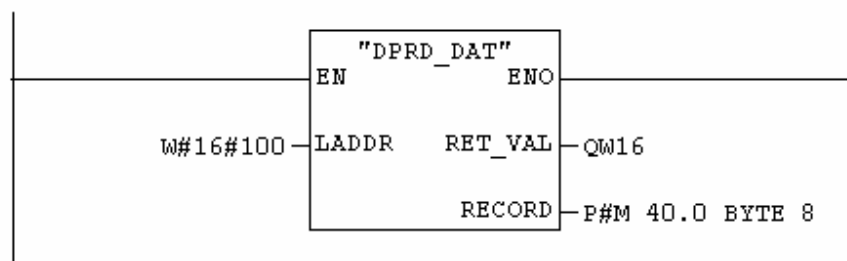
O comprimento dos dados deve ser até 3 bytes ou muito mais do que 4 bytes. O comprimento máximo depende da CPU. Você encontra esta informação na especificação técnica da sua CPU. Se nenhum erro ocorrer durante a transmissão, os dados lidos são inseridos na área de destino especificada pelo RECORD.

A área de destino deve ser do mesmo tamanho que a área que você tiver configurado para o módulo selecionado com o STEP 7.

Se um escravo DP padrão é de projeto modular ou tem diversos identificadores DP, você somente pode acessar os dados de um módulo/identificador DP neste instante no endereço de partida configurado com uma SFC 14 chamada.

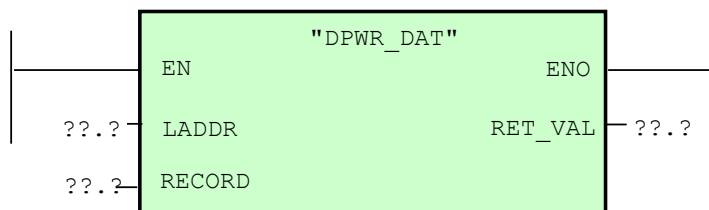
### Exemplo

**Network 1**: Read 4 analog values from starting address 256



## Escrevendo Consistência Dados dos Escravos DP Padrões c/ SFC 15

- Você necessita da SFC 15 "DPWR\_DAT" para escrever mais de quatro bytes consecutivos de dados (dados consistentes).



Parâmetro	Declaração	Tipo Dado	Área Memória	Descrição
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Configura endereço de partida na área de saída do módulo do qual o dado será escrito.
RECORD	INPUT	ANY	I, Q, M, D, L	Área fonte dos dados do usuário para escrita. Ela deve ser exatamente do mesmo tamanho que a área que você havia configurado para o módulo selecionado com o STEP7. Somente tipos de dados BYTE são permitidos.
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Se um erro ocorrer durante o processamento da função, o valor retornado contém um código de erro.

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.15Conhecimento em Automação  
Training Center

### Função

Com a SFC 14 "DPWR\_DAT" você escreve dados no arquivo (RECORD) consistentemente para o escravo DP padrão endereçado.

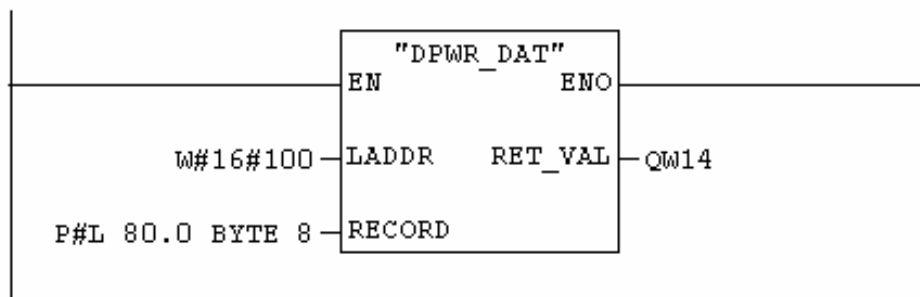
O tamanho dos dados devem ser quantificados para três ou mais de quatro bytes. O máximo comprimento depende da CPU. Você irá encontrar isto nas especificações técnicas da sua CPU. O dado é transmitido sincronamente, isto é, a escrita é completada quando a execução da SFC é terminada.

A área fonte deve ser do mesmo tamanho que a área que você tinha configurado para o módulo selecionado com o STEP 7.

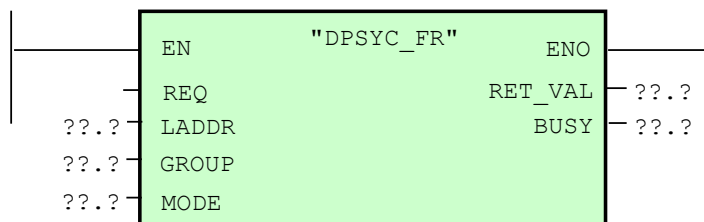
Se o escravo DP padrão é de projeto modular, você somente pode acessar um módulo de escravo DP.

### Exemplo

**Network 2:** Output analog values to address 256



## Sincronizando Escravos DP com SFC 11 (DPSYC\_FR)



Parâmetro	Declaração	Tipo Dado	Área Memória	Descrição
REQ	INPUT	BOOL	I, Q, M, D, L, Const.	Parâmetro de controle gatilhado por nível. REQ=1: Gatilho para tarefa SYNC/FREEZE.
LADDR	INPUT	WORD	I, Q, M, D, L, Const.	Endereço lógico do mestre DP.
GROUP	INPUT	BYTE	I, Q, M, D, L, Const.	Seleção de grupo, Bit 0 = 1: Grupo 1 selecionado Bit 1 = 1: Grupo 2 selecionado... Bit 7 = 1: Grupo 8 selecionado Você pode selecionar diversos grupos p/uma tarefa.
MODE	INPUT	BYTE	I, Q, M, D, L, Const.	Identificador de tarefa (de acordo com EN 50 170 V 3) Bit 0, 1, 6, 7: Reservado (valor 0) Bit 2 = 1: UNFREEZE é executado Bit 3 = 1: FREEZE é executado Bit 4 = 1: UNSYNC é executado Bit 5 = 1: SYNC é executado
RET_VAL	OUTPUT	INT	I, Q, M, D, L	Código de erro. Você deve avaliar RET_VAL após cada execução do bloco.
BUSY	OUTPUT	BOOL	I, Q, M, D, L	BUSY=1: A tarefa ainda não foi completada.

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.16Conhecimento em Automação  
Training Center

### Descrição

Com a SFC 11 "DPSYC\_FR", você pode sincronizar um ou mais grupos de escravos DP. Para fazer isto, você envia um dos seguintes comandos de controle ou uma combinação destes para os grupos desejados:

- SYNC (saída simultânea e congelamento do estado de saída dos escravos DP)
- UNSYNC (cancela o comando de controle SYNC)
- FREEZE (congelamento do estado da entradas do escravo DP leitura das entradas congeladas)
- UNFREEZE (cancela o comando de controle FREEZE)

### Pré-requisitos

Antes de você enviar os comandos de controle mencionados acima, você deve ter dividido os escravos DP em grupos SYNC ou FREEZE com o STEP 7.

### Qual é o efeito do SYNC?

Com o comando de controle SYNC, os escravos DP dos grupos denominados são chaveados para o modo Sync, isto é, o mestre DP transmite os dados de saída correntes e provoca nos escravos DP afetados o congelamento das saídas. Quando eles recebem as próximas mensagens de saída, os escravos DP simplesmente salvam os dados de saída em um buffer interno; o estado das saídas permanecem inalterados indefinidamente.

Após cada comando SYNC, os escravos DP dos grupos selecionados colocam seus dados de saída guardados no buffer simultaneamente nas saídas de periferia para o processo (saída simultânea pelo sinal de controle).

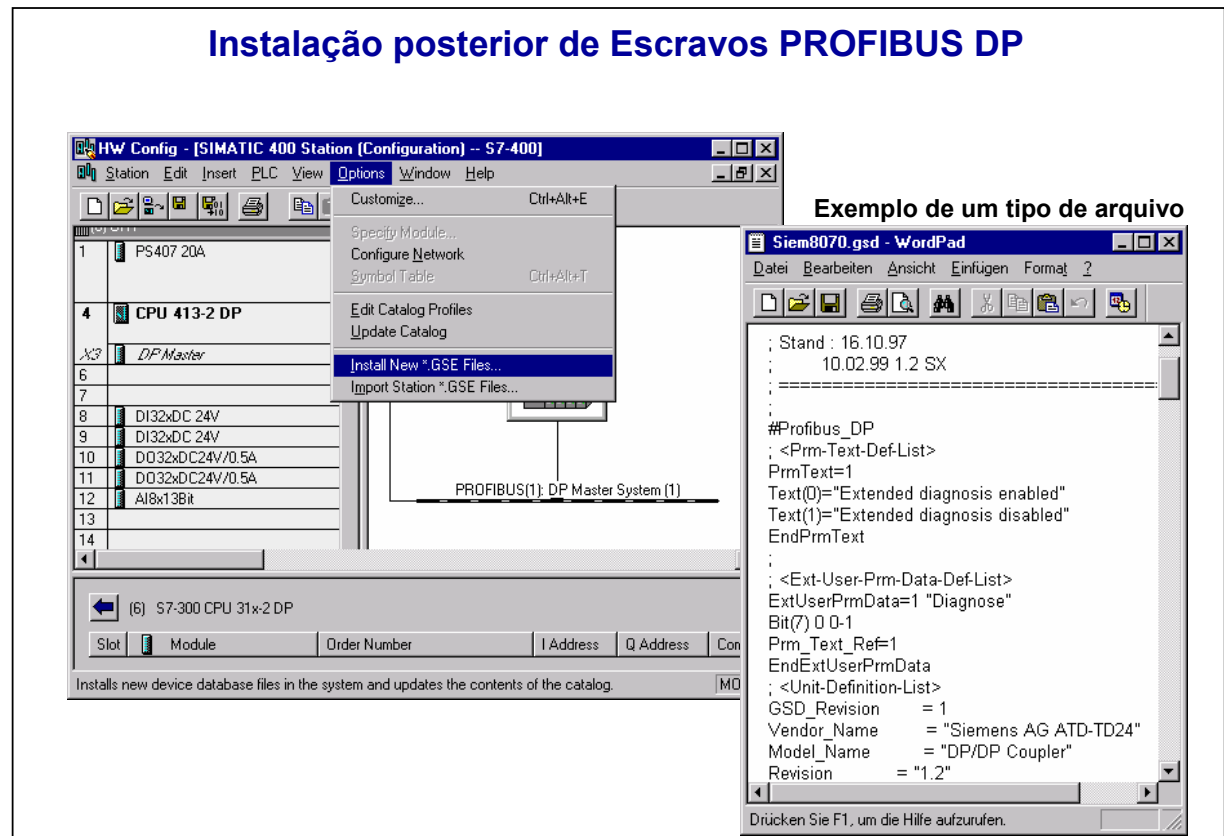
### Qual é o efeito do FREEZE?

Com o comando de controle FREEZE, os escravos DP afetados são chaveados para o modo Freeze. Cada comando FREEZE do mestre DP provoca nos escravos DP afetados a salvarem os estados correntes das entradas simultaneamente. Após isto, o mestre DP transmite os dados salvos para dentro de uma área de entrada da CPU.

As entradas ou saídas somente são atualizadas ciclicamente novamente quando o comando de controle UNSYNC ou UNFREEZE for enviado.



## Instalação posterior de Escravos PROFIBUS DP



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_12P.17Conhecimento em Automação  
Training Center

### Tipos de Arquivos

O STEP 7 necessita de arquivos de bancos de dados de equipamentos (GSD) ou arquivos tipo para cada escravo DP para que então você possa selecioná-lo do catálogo de hardware (Hardware Catalog) na ferramenta de configuração de hardware (HW Configuration).

Um arquivo GSD contém todas as propriedades do escravo DP de acordo com os padrões PROFIBUS. Os arquivos tipo são de acordo com as especificações Siemens.

Existe um arquivo tipo para cada tipo de escravo DP da SIEMENS AG. Um GSD ou arquivo tipo é fornecido com escravos DP de outros fabricantes.

### Integrando Escravos DP

Você pode integrar um novo escravo DP no catálogo de hardware conforme segue:

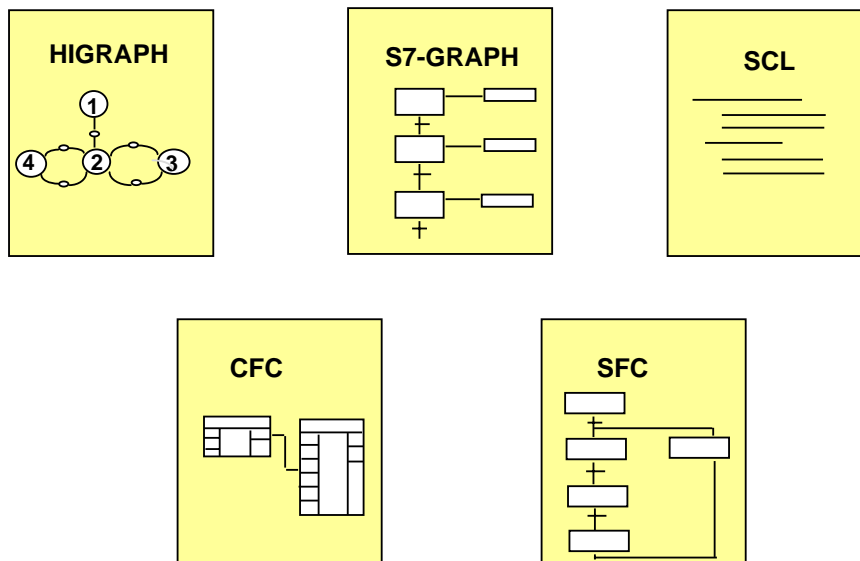
1. Selecione a opção de menu *Options -> Install new GSD*.
2. Na caixa de diálogo que se abre aparece o drive/diretório contendo os arquivos GSD existentes.

O escravo é inserido na janela "Hardware Catalog" (somente no perfil do catálogo "Standard") sobre "PROFIBUS-Additional Field Devices" e está disponível para configuração.

Quando os escravos DP são instalados ou importados deste modo, os arquivos GSD existentes e símbolos não são apagados completamente, mas são salvos em um diretório de backup \\Step7\\S7data\\Gsd\\Bkp[No.].

No. é um número de série o qual é fornecido automaticamente pelo STEP 7.

## Ferramentas de Engenharia para S7/M7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.1



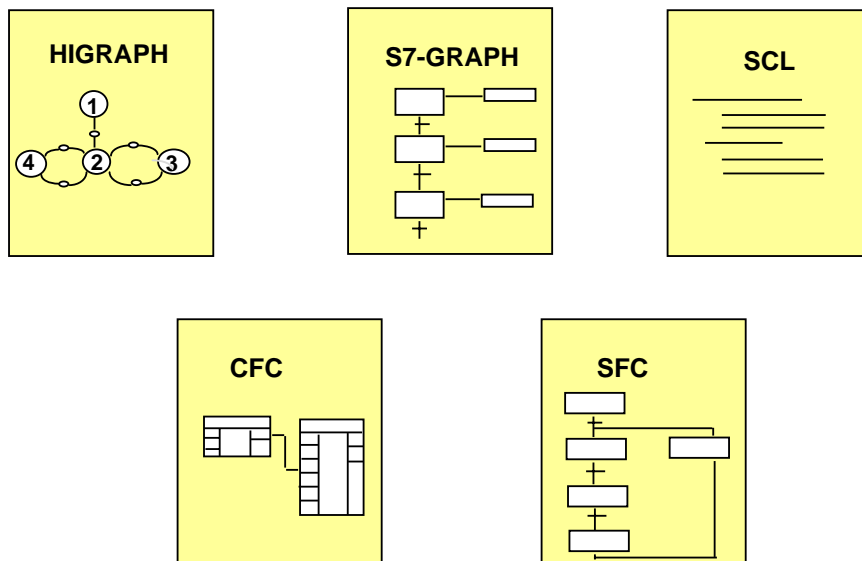
Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

O Pacote de Software S7- GRAPH .....	3
Estrutura de Programa de um Sistema de Controle Sequencial .....	4
Criando um FB Sequenciador .....	5
Visualizando Sequenciadores .....	6
Elementos de um Sequenciador .....	7
Programação de Ações .....	8
Ações Padrão em um Passo .....	9
Ações Dependentes de um Intertravamento .....	10
Ações Gatilhadas por um Evento .....	11
Verificando Condições em Transições, Intertravamentos e Supervisões .....	12
Instruções Permanentes .....	13
Criando um Bloco Executável .....	14
Integrando uma chamada de FB no OB1 .....	15
Ativação das Funções de Depuração (Debugging) .....	16
O Pacote de Software S7- HiGraph .....	17
Princípio do Método de Diagrama de Estados .....	18
Elementos de um Diagrama de Estados .....	19
Exemplo: Diagrama de Estados de um Controlador de Elevador .....	20
Criando um Diagrama de Estados .....	21
A Interface do Usuário HiGraph .....	22
Inserindo Estados e Transições .....	23
Programando Ações .....	24
Programando Transições .....	25
Programando Instruções Permanentes .....	26
Programando Grupos Gráficos (Graph) .....	27

## Ferramentas de Engenharia para S7/M7



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.2



Conhecimento em Automação  
Training Center

### Conteúdo

### Pág.

Atribuindo Parâmetros Atuais .....	28
Troca de Mensagens entre Diagramas de Estado .....	29
Atribuindo Valores Atuais para Mensagens .....	30
Salvando e Compilando .....	31
Depurando Funções em S7-HiGraph .....	32
Programando na Linguagem de Alto Nível S7- SCL .....	33
Estrutura de um Arquivo Fonte SCL .....	34
A Parte de Declarações de um Bloco .....	35
A Parte de Instruções do Bloco .....	36
Expressões, Operadores e Operandos em S7-SCL .....	37
Instruções em S7-SCL .....	38
Atribuição de Valores em S7-SCL .....	39
A Instrução IF em S7-SCL .....	40
A Instrução WHILE em S7-SCL .....	41
Chamando Blocos de Funções .....	42
O Flag "OK" para Avaliação de Erro .....	43
Compilando um Arquivo Fonte SCL .....	44
Monitoração Contínua .....	45
Setando e Editando Pontos de Parada (Breakpoints) .....	46
CFC para SIMATIC S7 e SIMATIC M7 .....	47
Gráficos CFC .....	48
Objetos CFC .....	49
Configurando Aplicações CFC em vez de Programação .....	50
Funções Integrais de Teste e Depuração .....	51
Configurando Sistemas de Controle Sequencial com S7-SFC .....	52
Cooperação entre CFC/SFC e SCL .....	53

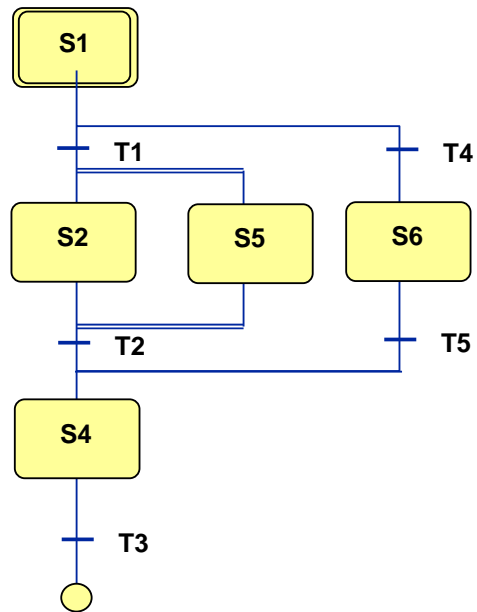
## O Pacote de Software S7- GRAPH

### S7-GRAPH: Ferramenta para programação de seqüenciadores

- Compatível com IEC 1131-3
- Projetada para necessidades da indústria de manufatura
- Interface Gráfica do processo em passos e transições
- Passos contendo ações
- Transições verifica as condições de habilitação do passo

### Você pode otimizar as seguintes fases em um projeto de automação com S7-GRAPH:

- Planejamento, configuração
- Programação
- Depuração
- Colocação em operação
- Manutenção, diagnóstico



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.3



Conhecimento em Automação  
Training Center

### S7-GRAPH

Com a linguagem de programação S7-GRAPH, você pode configurar limpa e rapidamente e programar seqüências que você deseja controlar com um sistema PLC S7.

O processo é dividido em passos simples com seus próprios escopos de funções. A seqüência é representada graficamente e pode ser documentada com figuras e textos.

As ações a serem realizadas são definidas em simples passos, as condições para movimentação de um passo para o próximo são definidos por transições. As definições destes, bem como os intertravamentos e supervisões são escritas em um subconjunto da linguagem de programação STEP 7 LAD (Diagrama Ladder). S7-GRAPH para S7-300/400 é compatível com a função seqüencial da linguagem gráfica definida no padrão IEC 1131-3.

### Funcionalidade

As seguintes funções são oferecidas:

- Diversos seqüenciadores (máx. 8) no mesmo bloco de funções S7- GRAPH
- Número livre de atribuições (1 a 999) para os passos (máx. 250 por seqüenciador complexo) e transições (máx. 250)
- Ramificações paralelas e ramificações alternativas (com um máx. de 250)
- Saltos (também para outros seqüenciadores)
- Partida/parada de seqüenciadores bem como ativação/suspensão de passos.

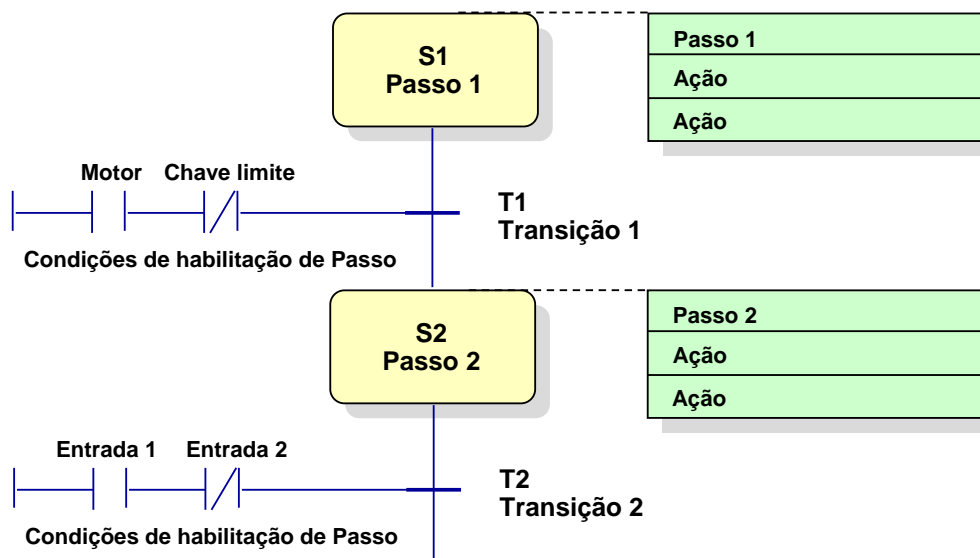
### Funções de Teste

- Mostra o passo ativo ou passo com falha
- Monitoração de estados e modificação de variáveis
- Chaveamento entre modos de operação: manual, automático e avanço

### Interface do Usuário

- Vista Geral, representação Página Simples Passo Simples.
- Distinção gráfica entre condições de intertravamento (intertravamento, máx. 32 condições), ações (máx. 100 por passo) e condições de supervisão (supervisão, máx. 32 condições)

## Estrutura de Programa de um Sistema de Controle Sequencial



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.4



Conhecimento em Automação  
Training Center

### Vista Geral

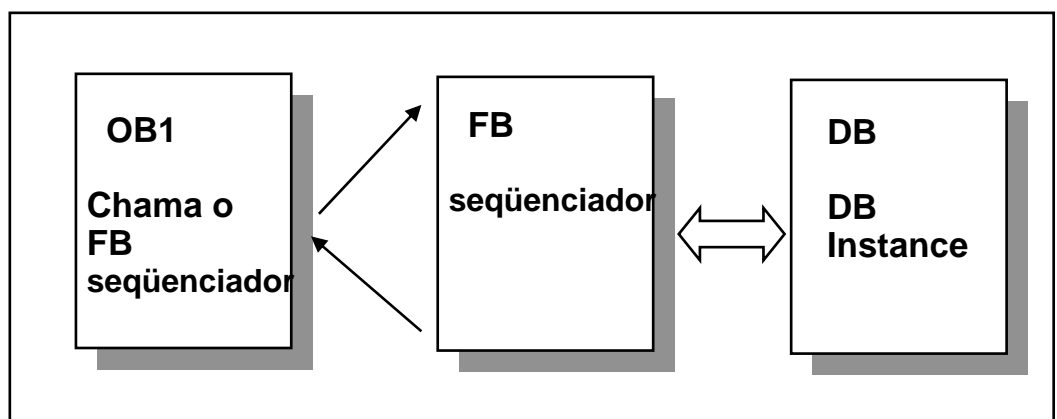
Um sistema de controle sequencial é um sistema de controle que é executado passo a passo. Este muda de um passo para o próximo passo quando as condições de habilitação forem atendidas. Uma característica do sistema de controle sequencial é a subdivisão das tarefas de controle em

- Passos e
- Transições (condições de habilitação de passos)

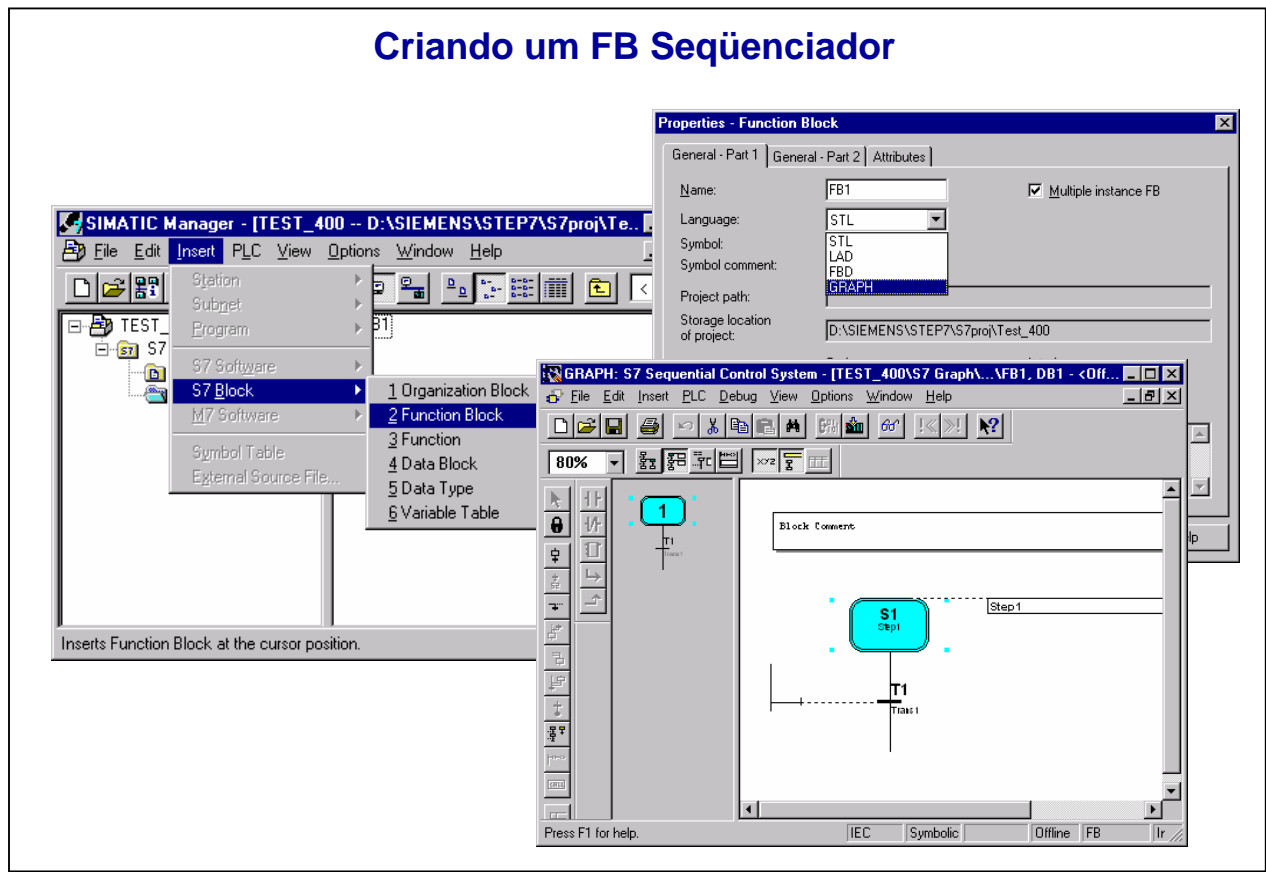
### Seqüenciador

Passos e transições formam um seqüenciador. O seqüenciador é guardado em um FB. Um DB instance, que contem os dados do seqüenciador, é atribuído a este FB. Pelo menos três blocos são necessários para um programa executável:

- o FB, que contem o seqüenciador(es)
- um DB instance com os dados do seqüenciador
- um bloco de organização, função ou bloco de funções contendo a chamada do FB. Os parâmetros e o número do DB instance são passados na chamada do FB.



## Criando um FB Sequenciador



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.5



Conhecimento em Automação  
Training Center

### Vista Geral

Você pode criar FBs S7-GRAPH com o gerenciador SIMATIC ou com o editor S7-GRAPH. Em ambos os casos, você deve primeiro criar um projeto e um correspondente programa do usuário.

### Criando um FB

Para configurar um FB S7-GRAPH com o gerenciador SIMATIC, proceda como a seguir:

1. Abra o programa do usuário no qual o FB será inserido.
2. Selecione o comando de menu *Insert -> S7 Block -> Function Block*
3. Na caixa de diálogo "Properties" insira o número do FB (p.ex. FB1) e insira GRAPH como sendo a linguagem de programação. Confirme com "OK"
4. Duplo clique para inserir o bloco. O editor S7-GRAPH é chamado e o bloco é aberto.

### Editor S7-GRAPH

Na inserção e edição de um seqüenciador, você será amparado pelo Editor S7-GRAPH com funções sensíveis ao contexto. A barra de ferramentas contém ícones que dão a você rápido acesso a comandos freqüentemente utilizados do menu.

As seguintes barras de ferramentas podem ser selecionadas através de comando de menu *View -> Toolbar*:

Standard: Contem funções para manipulação de arquivos (Abrir, Salvar, etc.) e para edição (Copiar, Inserir, etc.).

Sequencer: Contem funções que inserem elementos seqüenciadores no programa.

LAD: Contem funções que inserem elementos LAD no programa.

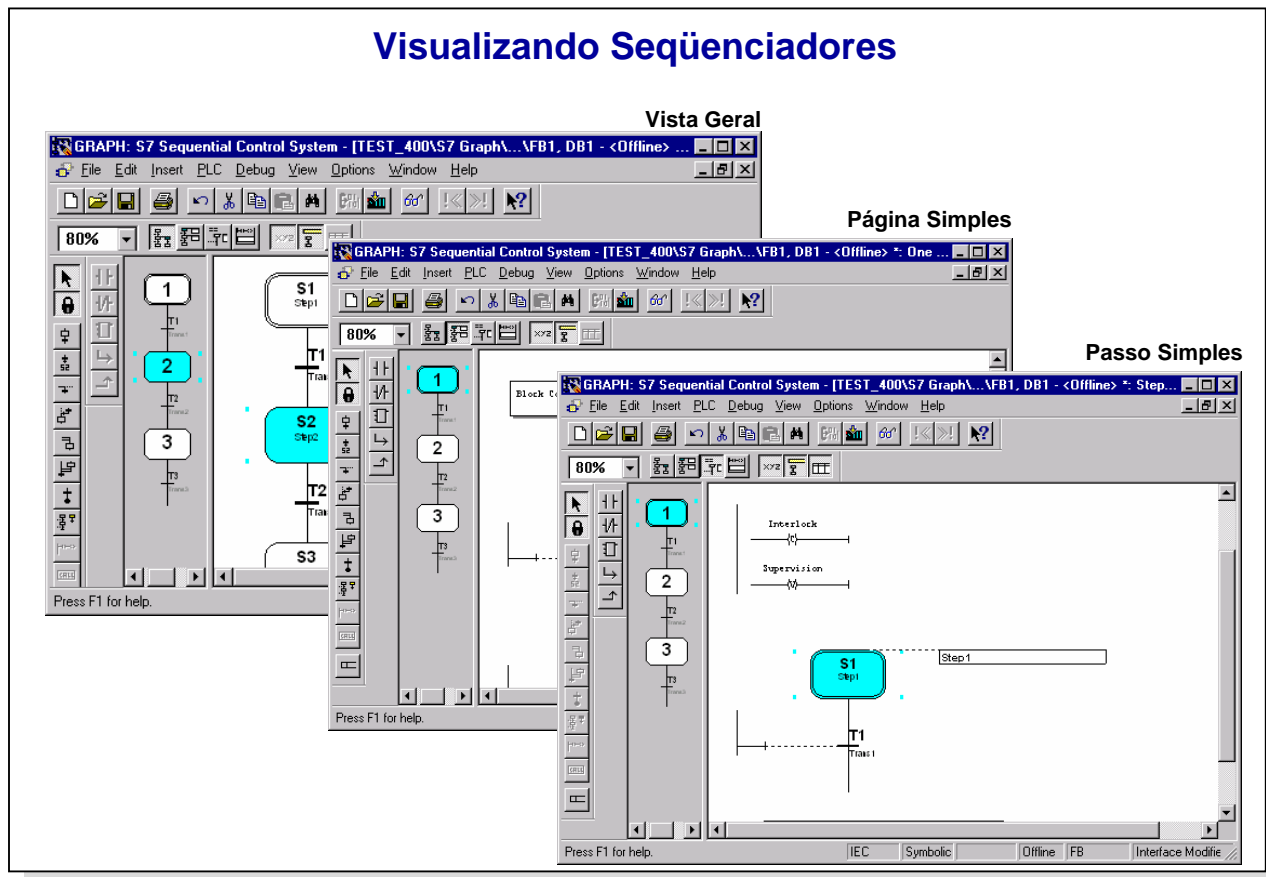
View: Contem funções para mudança do modo de visualização

### Nota

Você pode posicionar individualmente as barras de ferramentas em qualquer lugar dentro do Editor S7-GRAPH.

Para isto, clique na área cinza da barra de ferramentas e arraste a barra para a posição desejada com o mouse.

## Visualizando Seqüenciadores



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.6



Conhecimento em Automação  
Training Center

#### Vista Geral

O Editor S7-GRAPH permite você editar um seqüenciador com todos os seus elementos, tais como passos, transições, ramificações, saltos, fim do seqüenciador e comentários, em diferentes vistas.

Você pode selecionar a vista pela escolha do seguinte comando de menu:

*View -> Overview/Single Page/Single Step/Permanent Instructions*

ou da barra de ferramentas.

#### Vista Geral

Esta vista dá a você uma impressão geral do sistema de controle seqüencial como um todo. Esta é especialmente disponível para configuração da estrutura seqüencial.

Diversos seqüenciadores (máx. 8) em um FB aparece lado a lado. A Vista Geral contém os comentários do bloco, nomes e números dos passos e transições.

#### Página Simples

Página Simples mostra ações, os números e os nomes dos passos, as condições de habilitação dos passos para as transições, os nomes e números das transições e os comentários do bloco.

Diversos seqüenciadores em um FB aparecem um debaixo do outro. Esta vista é disponível para configuração e para programação de um sistema de controle seqüencial.

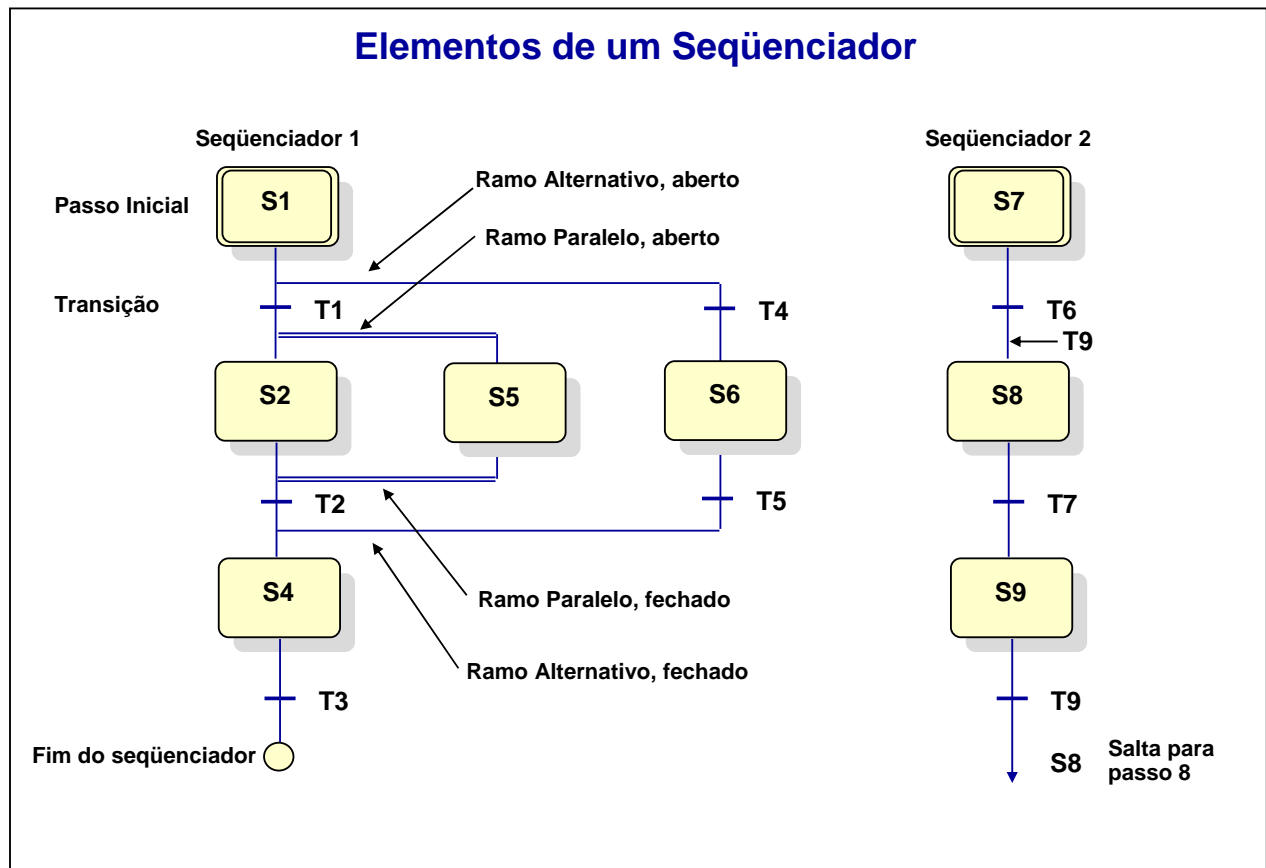
#### Passo Simples

Esta vista mostra um passo com suas transições e todo seu conteúdo. Esta vista é usada para programação não somente de ações e condições de habilitação de passo mas também supervisões e intertravamentos. Você também pode editar comentários de passos nesta vista.

#### Instruções Permanentes

Você usa esta janela para programação de "instruções permanentes". Instruções permanentes são condições e/ou chamadas de blocos as quais vem antes ou depois do seqüenciador. Elas são executadas uma vez em cada ciclo independentemente do estado do seqüenciador.

## Elementos de um Seqüenciador



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.7Conhecimento em Automação  
Training Center

### Vista Geral

Quando escrevendo um seqüenciador, você está amparado por funções de programação gráfica. Você pode gerar uma estrutura seqüencial simplesmente sem necessitar de extensa experiência em programação, pelo arranjo de elementos na estrutura S7-GRAPH produz uma representação gráfica do seqüenciador.

### Estrutura de um Seqüenciador

Um seqüenciador consiste de uma série de passos e transições. O seqüenciador pode ser linear ou ramificado.

- Dentro dos passos, você formula as instruções para a planta.
- As transições contêm as condições para mudança de um passo para o próximo.

Os seguintes ícones representam os elementos que podem constituir um seqüenciador. Você pode selecionar estes ícones da barra de ferramentas.



Também existem ícones para inserção de instruções permanentes e para mudança do modo de inserção.

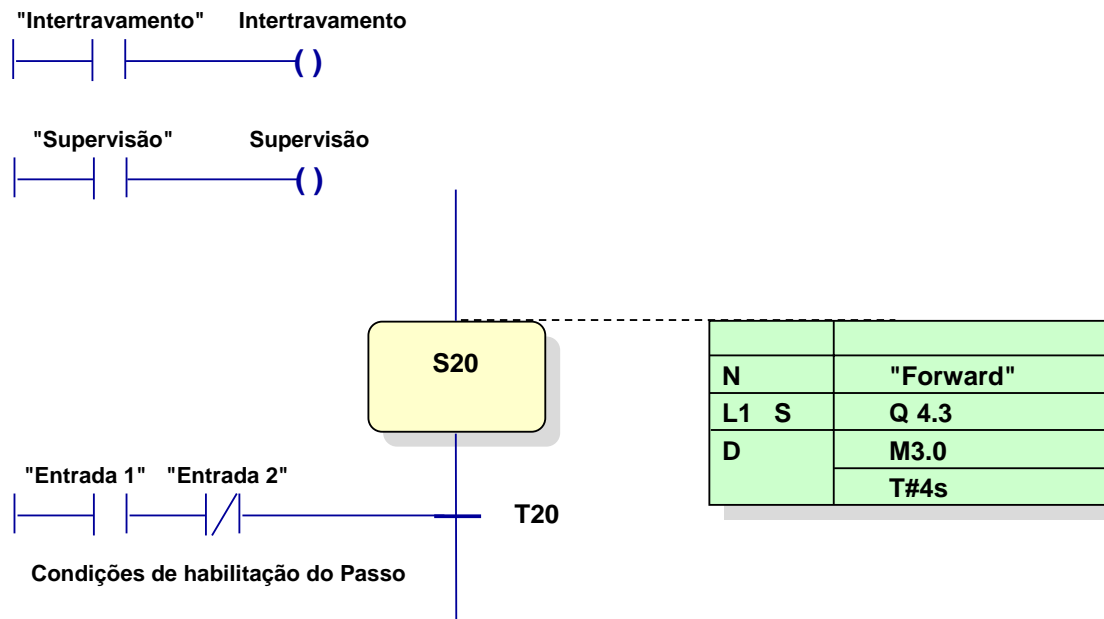
### Regras

A estrutura do sistema de controle seqüencial deve satisfazer as seguintes regras:

- Passos e transições somente podem ser inseridos aos pares.
- Saltos podem ser adicionados ao fim de um ramo e saltar para um passo no mesmo seqüenciador ou em um seqüenciador diferente.
- Um símbolo de fim de seqüenciador pode ser adicionado seguindo uma transição para o fim de um ramo e terminar a execução do ramo.



## Programação de Ações



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.8Conhecimento em Automação  
Training Center

### Vista Geral

As partes ativas do passo são ações que são executadas pelo controlador no passo ativo. Nos passos, instruções são programadas as quais configuram saídas ou ativam/desativam passos no seqüenciador. Uma ação consiste de instruções e endereços e podem ser anexadas a condições e/ou combinadas com um evento.

### Programação

Você programa ações em passos simples ou páginas simples. Para fazer isto, proceder como abaixo:

1. Selecione a tabela da direita do passo e pressione a tecla Tab.
2. Agora programe as ações pela inserção de instruções as quais estão sendo executadas na tabela:
  - Cada instrução ocupa exatamente uma linha da tabela
  - Na coluna da esquerda estão as instruções, na coluna da direita estão os endereços
  - Se você usar uma instrução a qual requer informação de tempo (D ou L), S7-GRAH automaticamente configura duas linhas na coluna da direita. Insira a informação de tempo na linha de baixo.
3. Press the Tab key once again to program a further action. The table is expanded by one line.

Se você tiver usado uma instrução que é logicamente combinada com uma condição (todas as instruções que contem a letra C), então você deve programar a condição na vista passo simples como um intertravamento.

### Intertravamento

A lógica de intertravamento é usada para habilitação condicional da ação específica em um passo. Se a condição for satisfeita, então todas as ações condicionadas com "C" são executadas. O avanço para o próximo passo ocorre independente da condição de intertravamento.

### Supervisão

A lógica de supervisão é usada para reconhecer erros de supervisão e revisão de reações (parando o seqüenciador e possivelmente reconhecendo obrigações). O avanço para o próximo passo somente ocorre quando as condições de supervisão não forem satisfeitas e nenhum erro tenha ocorrido.

## Ações Padrão em um Passo

### Bloco ação com instruções simples

Action_block_1	
N	M1.1
S	M1.2
R	M1.3
D	M1.4
	T#1H2M3S
L	M1.5
	T#4MS
CALL	FC1

- N = Atribuição Não armazenada
- S = Seta (Stored)
- D = Time Delayed, atributo não armazenado atrasado pelo tempo T
- L = Time Limited, atributo não armazenado para um tempo limitado
- CALL = chamada de Bloco

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.9



Conhecimento em Automação  
Training Center

<b>Vista Geral</b>	Estas ações são todas definidas no padrão IEC 1131-3 sobre 2.6.4.
<b>D, L, N</b>	As ações associadas são resetadas tão logo o passo seja completado.
<b>D, L</b>	Os tempos podem ser especificados como uma constante ou uma variável.
<b>S, R</b>	A ação correspondente mantém ativa mesmo após a execução do passo.
<b>CALL</b>	Chama blocos FBi.DBi, FCi, SFBi.DBi, SFCi: chama os blocos especificados. Após o processamento do bloco, o programa GRAPH é executado na sequência.
<b>Nota</b>	Os endereços no bloco de ação pode ser simbólico ou absoluto.
<b>Times</b>	<p>Os tempos (times) devem ser inseridos no formato de tempo IEC. Este é como abaixo:</p> <p>T#mDnHoMpSqMS</p> <p>mD: m Dias nH: n Horas oM: o Minutos pS: p Segundos qMS q Milisegundos</p> <p>O máximo tempo é limitado a aproximadamente 24D20H.</p>

## Ações Dependentes de um Intertravamento

### Bloco ação com instruções condicionais

Action_block_2	
NC	M1.1
SC	M1.2
RC	M1.3
DC	M1.4
	T#1H2M3S
LC	M1.5
	T#4MS
CALLC	FB5.DB3

### Condições

- Uma ação identificada com um "C" (Condição) somente é executada se a condição de intertravamento do passo é verdadeira ("C" = 1).
- Um erro de intertravamento existe se a condição é zero. A ação sujeita a condição C é então não executada. O passo é marcado e a mensagem de erro "Error" é fornecida.

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.10Conhecimento em Automação  
Training Center

### Caso Especial

Uma ação em um passo o qual tem o qualificador "C" mas para o qual nenhum intertravamento está programado é executado incondicionalmente. O qualificador "C" é ignorado.

### Exemplos:

Instrução	Endereços	Explicação
NC	Q1.0	Enquanto o passo estiver ativo e a condição estiver satisfeita, o sinal em Q1.0 = 1, caso contrário 0.
SC	Q1.0	Enquanto o passo estiver ativo e a condição estiver satisfeita, Q1.0 = 1, e se mantém em 1.
RC	Q1.0	Enquanto o passo estiver ativo e a condição estiver satisfeita, Q1.0 = 0 e se mantém em 0.
DC	Q1.0 T#<const>	Após o fim do tempo especificado na <const> e enquanto o passo estiver ativo e a condição estiver satisfeita o sinal em Q1.0 é = 1. Se o passo não estiver ativo o sinal é = 0.
LC	Q1.0 T#<const>	Enquanto o passo estiver ativo e a condição estiver satisfeita, o sinal em Q1.0 é = 1 para o tempo especificado <const>. Se o passo não estiver ativo, o sinal é = 0.
CALLC	FB5.DB3	Chama o bloco especificado, quando a condição estiver satisfeita. Após o processamento do bloco, o programa GRAPH é executado.

## Ações Gatilhadas por um Evento

### Bloco ação com instruções dirigidas a evento

Action_block_3		
A1	N	M1.1
L1	N	M1.2
L0	N	M1.3
S1	N	M1.4
S0	N	M2.4
V1	N	M2.5
V0	N	M2.6

A ação é executada uma vez no ciclo no qual o evento ocorre

- A1 = Reconhecimento
- L1 = Erro de intertravamento chegando
- L0 = Erro de intertravamento indo
- S1 = Passo ativado
- S0 = Passo desativado
- V1 = Erro de supervisão chegando
- V0 = Erro de supervisão indo

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.11



Conhecimento em Automação  
Training Center

### Vista Geral

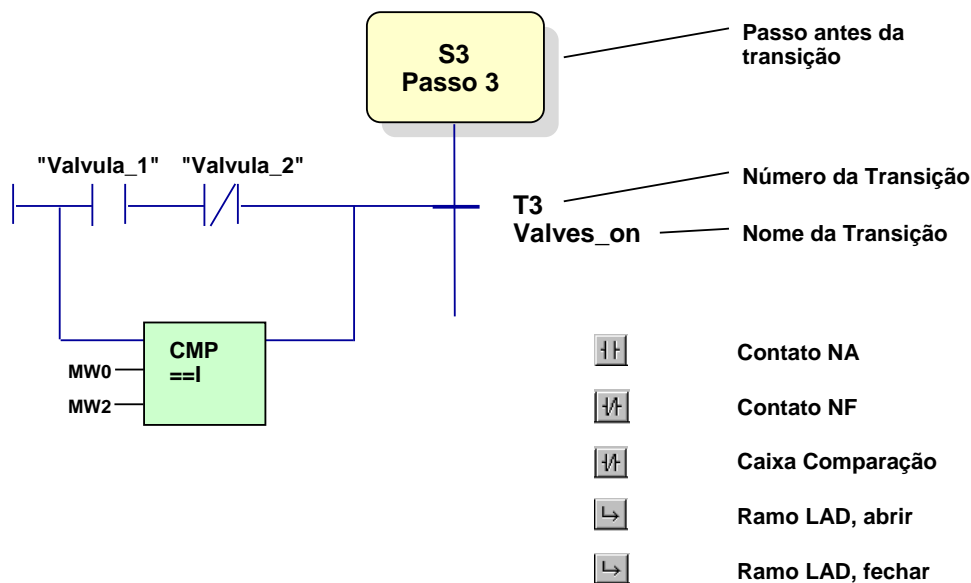
Um evento pode ser detectado e ligado com uma ação. Deste modo é possível não somente monitorar passos individuais mas também monitorar e influenciar o sistema de controle seqüencial inteiro.

### Ativando e Desativando Passos

Outros passos de um seqüenciador podem ser ativados ou desativados com as instruções ON e OFF.

Evento	Instrução	Endereço	Explicação
S1 S0 V1 V0	ON  OFF	Si  i=Número Passo	Dependendo do evento, ativa ou desativa o passo
L1 L0 A1	OFF	S_ALL	Dependendo do evento, ativa ou desativa todos os passos; exceto para o passo que contém a ação.

## Verificando Condições em Transições, Intertravamentos e Supervisões



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.12



Conhecimento em Automação  
Training Center

### Vista Geral

Você programa uma condição como uma operação lógica booleana na forma de elementos em Diagrama Ladder. Para cada transição, intertravamento e supervisão existe um segmento (network) LAD na qual você pode inserir elementos LAD.

### Condições

O S7-GRAPH reconhece os seguintes tipos de condições:

Transition: (transição) descreve as condições de habilitação do passo pela qual um controlador passa de um passo para o passo seguinte.

Permanent Condition: condições permanentes são arranjadas antes ou depois do seqüenciador e são avaliadas uma única vez por ciclo.

Interlock: (intertravamento) descreve condições que devem ser satisfeitas para permitir que uma ação seja executada. Um erro de intertravamento é sinalizada se o intertravamento do passo for perdido.

Supervision: (supervisão) descreve condições que levam um erro de supervisão sendo sinalizado e que bloqueia o seqüenciador a mudar para o próximo passo.

### Elementos LAD

Os seguintes elementos estão disponíveis para verificação de condições: contato NA, contato NF, caixa de comparação, abrir e fechar ramificação LAD.

### Edição

É mais fácil programar as condições na vista passo simples. Você pode escolher como você deseja usar os ícones na barra de ferramentas para edição de um seqüenciador pela seleção da opção de menu

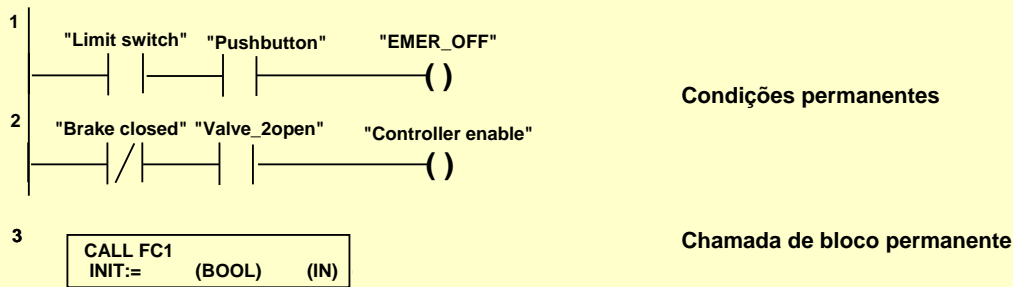
*Insert* -> *Drag&Drop/Append* ou pelo clique no ícone correspondente na barra de ferramentas:

Drag&Drop or Insert Mode ON: primeiro selecione o ícone na barra de ferramentas e então clique com o mouse no lugar onde você deseja inserir o objeto.

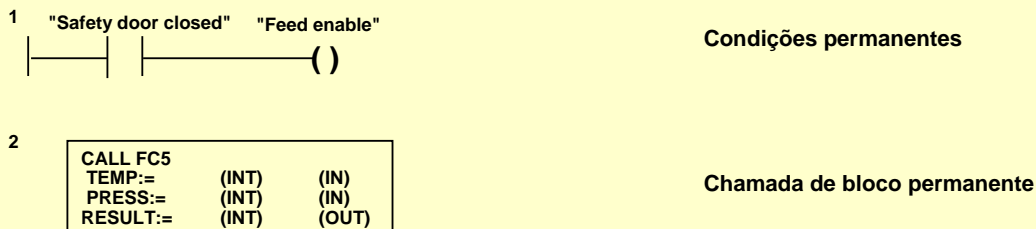
Append or Insert Mode OFF: primeiro selecione o elemento no segmento (network) após o qual você deseja inserir um novo elemento e então clique no ícone desejado da barra de ferramentas.

## Instruções Permanentes

### Instruções permanentes antes do seqüenciador



### Instruções permanentes após o seqüenciador



## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.13



Conhecimento em Automação  
Training Center

### Vista Geral

Instruções permanentes são condições ou chamadas de bloco localizada antes ou depois do seqüenciador e executado uma vez por ciclo.

Você pode usar instruções permanentes, por exemplo, para:

- Programar somente uma vez condições que necessitem ser satisfeitas em diversos pontos no seqüenciador.
- Chamada de blocos que contenham instruções STL, LAD, FBD ou SCL do S7-GRAPH.

Você pode ter quantas instruções permanentes em um FB GRAPH quantas você queira.

### Instruções Permanentes

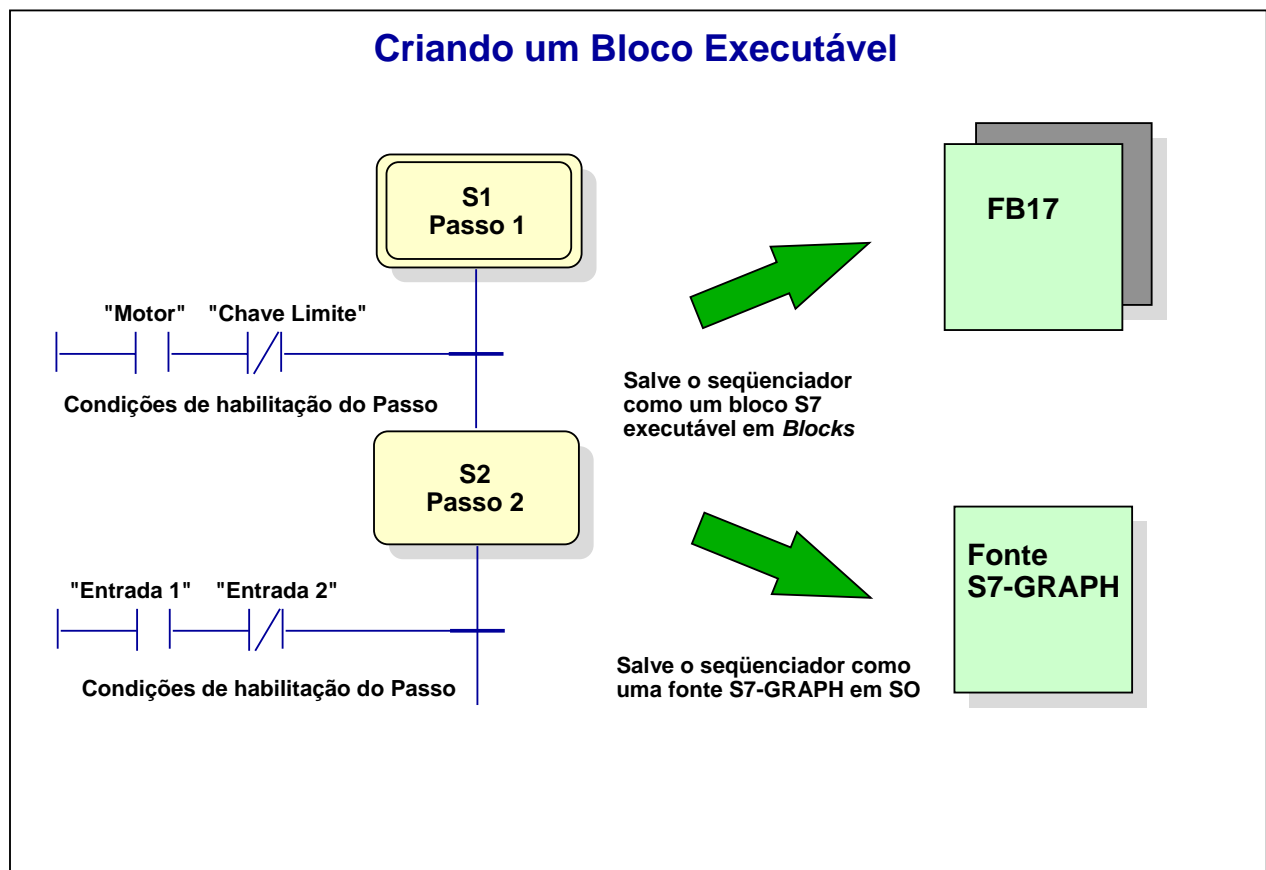
Você programa condições permanentes ou chamadas de blocos na vista *Permanent Instructions*. Selecione o modo de inserção marque&arraste com a opção de menu *Insert -> Drag&Drop* e proceda como a seguir:

1. Selecione a opção de menu *View -> Permanent Instructions*, se a janela necessária não estiver ainda sendo mostrada.
2. Escolha a opção de menu *Insert -> Permanent Instruction- > Condition/Call* ou clique com o mouse no ícone correspondente na barra de ferramentas.

O ponteiro do mouse seu formato para representar um segmento (network) em LAD ou uma instrução CALL.

3. Localize o ponteiro do mouse no espaço fornecido.
4. Escolha a opção de menu ou clique o ícone da barra de ferramenta novamente.
5. Insira os elementos LAD que você deseja usar para programar as condições ou insira o nome do bloco que você deseja chamar após a instrução CALL e pressione a tecla Enter.
6. No caso de um bloco chamado, atribua parâmetros atuais para os parâmetros formais mostrados.

## Criando um Bloco Executável



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.14



Conhecimento em Automação  
Training Center

### Vista Geral

Quando se salva um bloco de funções S7-GRAPH, a função de compilação é gatilhada. Isto é, um bloco de funções que pode ser transferido para o PLC é criado de um seqüenciador. Somente programas do usuário livre de erros podem ser salvos. Se o programa não podem ser compilados devido a erros então ele também não pode ser salvo.

Se você tiver de interromper o seu trabalho o qual ainda contenha erros de configuração, você pode a qualquer tempo salva-lo no estado presente em um arquivo fonte S7-GRAPH pela utilização do comando de menu *File -> Generate Source*.

### Opções

Você pode setar as seguintes opções para compilação pela escolha do item de menu *Options -> Customize* e então a tabela de compilação:

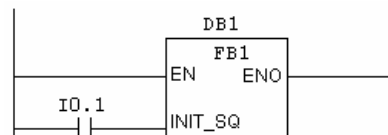
- Quaisquer parâmetros estão disponíveis quando você chama os FBs S7-GRAPH (mínimo, padrão ou conjunto máximo de parâmetros)
- Como os passos e transições estão sendo armazenados em um DB instance (como estrutura de arrays ou como estruturas individuais) e se esta descrição de interface está sendo transferida para o PLC ou não.
- Se o FB gerado está sendo executado em si próprio ou se uma FC padrão (FC 70 or FC 71) contendo a maior parte do código gerenciador está sendo usado. Se você escolher a opção FC padrão, você pode armazenar mais passos no FB então se ele está sendo executado em si próprio.
- Se dados de análise de condições estão sendo escritos no DB instance, se o modo Skip estiver ativado e se erros de supervisão ocorreram durante a operação tiverem de ser reconhecidos.
- Se programas e processos estão sendo sincronizados e se todos as condições de intertravamento já estão sendo processadas em operação. Quando mostrando os estados, um intertravamento perdido e o passo o qual esteja com falha como uma consequência estão então mostrados.
- Se mensagens de alerta estão sendo mostradas na janela de mensagem durante a compilação.
- Se erros de intertravamento e supervisão estão sendo manipulados pela SFC 52 (entrada do buffer de diagnóstico) ou usando a SFC 17 e 18 (envio para equipamentos IHM).

## Integrando uma chamada de FB no OB1

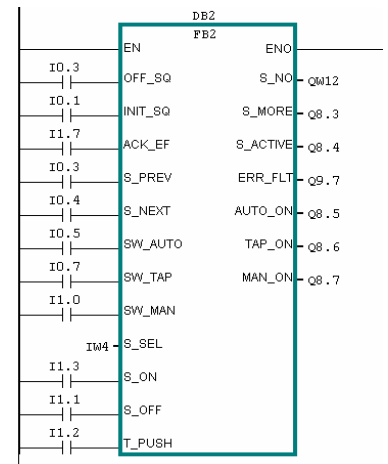
### Duas formas de chamada de bloco

- **Parâmetros mínimos de chamada (padrão)**
  - 1 parâmetro de entrada para controle do seqüenciador
- **Parâmetros padrões de chamada**
  - 12 parâmetros de entrada para controle do seqüenciador
  - 7 parâmetros de saída para mostrar estados de operação
- **Conjunto máximo de chamada**
  - 17 parâmetros de entrada para controle do seqüenciador
  - 12 parâmetros de saída para mostrar estados de operação

Conjunto mínimo de parâmetros



Conjunto máximo de parâmetros



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.15Conhecimento em Automação  
Training Center

#### FB Seqüenciador

Pela seleção da opção de menu *Options -> Cutomize* e a página tabela de Compilação você também pode escolher se o FB está sendo atribuído o conjunto de parâmetros mínimo, padrão ou máximo quando é chamado.

O FB responde a bordas de subida em seus parâmetros de entrada.

#### Parâmetros Mínimos

O FB com conjunto de parâmetros mínimos de entrada tem somente um parâmetro de entrada, INIT\_SQ, e ativa seu seqüenciador tão logo quanto tenha sido processado no OB1. Isto significa que os seqüenciadores são executados imediatamente em modo automático.

Você usa o use conjunto de parâmetros mínimos quando você executa o seqüenciador somente em modo automático e quando você não necessita de qualquer função adicional de monitoração e modificação.

O passo(s) inicial(is) é/são ativados por borda de subida no parâmetro INIT\_SQ.

#### Parâmetros Padrão

O modo de operação também deve ser selecionado quando chamando este FB. Você sempre utiliza o conjunto de parâmetros padrões quando o seqüenciador está operando em modos de operação diferentes e quando você requer informações de retorno sobre o processo e facilidades para reconhecimento de mensagens.

O FB seqüenciador sempre mantém o último modo de operação ativado. O modo de operação anterior somente pode ser deselecionado pela seleção de um diferente. Parâmetros que não são requeridos mantem-se desatribuídos.

#### Parâmetros Máximos

Você sempre utiliza o conjunto máximo de parâmetros quando você necessita de maior intervenção do operador e facilidades de monitoração para serviços e colocação em operação estes então fornecidos pelo conjunto de parâmetros padrão.

Todos os parâmetros do FB são mostrados e podem ser atribuídos na chamada do FB (requer maior capacidade de memória).



## Ativação das Funções de Depuração (Debugging)

### Procedimento

- **Transferência do FB seqüenciador e DB instance**
  - Utilize o comando de menu **PLC -> Download** para transferir o FB seqüenciador e o DB instance para o PLC
- **Selecione o DB instance:**
  - Selecione o DB instance que você deseja utilizar para testar pela escolha do comando de menu **Debug -> Test Environment**
- **Inicialize a função "Monitor":**
  - Selecione a seção desejada do sistema de controle seqüencial. A informação de estado irá fornecer a parte visível corrente na janela aberta.
  - Ative o comando de menu **Debug -> Monitor** (marcas de verificação)
- **Saia da função "Monitor":**
  - Desative o comando de menu **Debug -> Monitor**

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.16



Conhecimento em Automação  
Training Center

### Funções de Depuração

Você pode executar o seqüenciador S7-GRAPH no modo teste com as funções de depuração. O estado dos elementos seqüenciadores e o estado do sinal dos endereços são mostrados na tela.

### Downloading to PLC

De forma a transferir o FB S7-GRAPH com o DB instance associado para o PLC, proceda como a seguir:

1. Com o FB aberto, selecione a opção **PLC -> Download**.
2. Na caixa de diálogo "Download" selecione o DB instance o qual você deseja transferir para a CPU junto com o DB aberto.
3. Se necessário, confirme os blocos serão sobrescritos com o mesmo nome dos já existentes na CPU.

### Nota

Se possível, transfira os blocos em modo STOP, devido a condições de erro que podem ocorrer se você sobrescrever um programa velho no modo RUN.

### Ativar Monitoração

Esta função é realizada simultaneamente para todas as janelas abertas do mesmo FB seqüenciador. Se muito mais informações de estados do S7 tem sido atualizadas, uma mensagem de alerta deve aparecer informando que agora nem todos os estados podem ser atualizados sincronizadamente.

Pelo reconhecimento da mensagem, você pode retornar o modo Monitor do seqüenciador.

### Desativar Monitoração

Monitoração do seqüenciador sempre deve ser interrompida antes de você fazer alterações no seqüenciador ou nos operandos lógicos. Somente então estas ações são possíveis no seqüenciador.

É recomendável que suas alterações sejam salvas primeiro no disco rígido e então transferidas para a CPU S7. Se você isto em ordem errada, uma janela de mensagem aparece, a qual aponta para a seqüência correta.

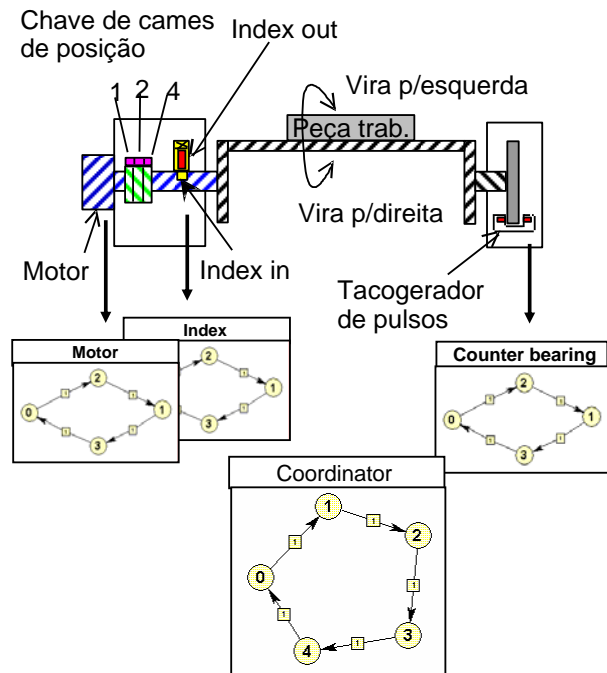
## O Pacote de Software S7- HiGraph

### S7-HiGraph: Ferramenta para programação com diagramas de estado

- A máquina é dividida em funções unitárias
- Diagramas de estado são criados para cada unidade de função
- Estados contêm ações
- Diagramas de estado comunicam por meio de mensagens

### Você pode otimizar as seguintes fases em um projeto de automação com S7-HiGraph:

- Planejamento, configuração
- Programação e depuração
- Colocação em operação
- Manutenção, diagnósticos
- Permite reutilização



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.17



Conhecimento em Automação  
Training Center

### Motivação para Diagramas de Estado

A construção de maquinário e plantas é um campo extenso no qual o controle de movimentos mecânicos assíncronos e tempos de respostas representam os principais problemas.

A linguagem mais comumente utilizada neste campo até o momento é STL e algumas vezes LAD, FBD e linguagens de seqüência (GRAPH5, S7-GRAPH, etc.). Contudo, estas linguagens não são particularmente aplicáveis para engenheiros mecânicos, por exemplo para formulação de aplicações de construção mecânica.

O resultado é que cada grupo, por exemplo os engenheiros mecânicos ou engenheiros de automação, utilizam seus próprios métodos (linguagens), as quais se tornam difíceis para troca de informações.

O auxílio para um grupo de trabalho (aprox. 1980) com a tarefa de investigação de caixas de ferramentas para PLCs foi especificar uma ferramenta que pudesse ser utilizada em todas as fases de um projeto desde a fase de definição, programação e manutenção. Esta ferramenta foi estar disponível para uso em todas as áreas e permitir uma significativa vantagem para solução de problemas de automação. Também foi um requisito que a documentação e programas devam ser reutilizáveis para outros projetos de tipos similares.

O resultado foi o método de diagramas de estado descrito aqui, o qual é colocado no mercado pela SIEMENS sobre o nome de produto S7-HiGraph. O S7-HiGraph pode ser utilizado em PLCs da série S7-300 (CPU 314 e superior) e S7-400.

### Vantagens

O formato orientado a objeto do S7-HiGraph é idealmente disponível para:

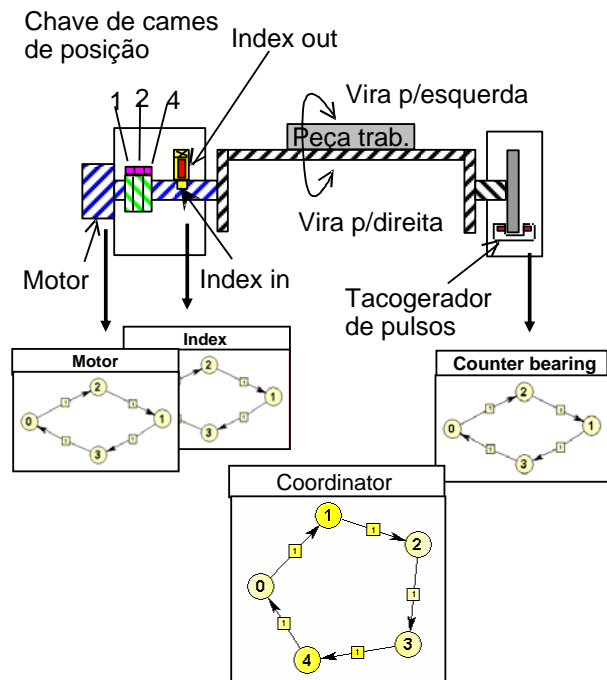
- Engenheiros de máquinas e plantas (projeto mecânico)
- Especialistas de automação (engenheiros eletricitas) como um significado comum da descrição
- Engenheiros de comissionamento e manutenção

O método de diagramas de estado habilita processos inteiros de construção de uma máquina ou planta se otimizada pela redução do tempo de desenvolvimento e revisões de projeto, bem como tempo de comissionamento.

## Princípio do Método de Diagrama de Estados

### Exemplo: Tabela Rotativa para máquina de acabamento

- **Unidades Funcionais (Function units) (FUs)**
  - Motor
  - Index
  - Tacogerador de pulsos
- **Diagramas de Estado**
  - Um diagrama p/cada FU
  - Um diagrama coordenador



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.18



Conhecimento em Automação  
Training Center

### Vista Geral

O método de diagramas de estado orientam-se através de "objetos" do mundo real quando utilizados para esboço e programação de projetos de automação. A máquina ou planta a ser automatizada é vista como uma combinação de elementos individuais ou unidades funcionais (function units).

### Unidades Funcionais

A unidade funcional (function unit) (FU) é a menor unidade mecânica de uma máquina ou planta. Uma FU é normalmente composta de elementos básicos mecânicos e elétricos.

Durante a programação, cada unidade funcional é atribuída a um diagrama de estado (ou graph) no qual as propriedades funcionais, que são mecânicas e elétricas, da FU são representadas.

Para o método de diagramas de estado o objeto a ser automatizado é quebrado em diversas partes menores ou unidades funcionais.

### Diagrama de Estado

O diagrama de estado descreve a dinâmica pertencente a unidade funcional. Ele descreve os estados que uma unidade funcional pode assumir e as transições entre eles.

Os diagramas de estado são seções de programa que podem ser usadas de novo e de novo. Os diagramas de estado criados para uma unidade funcional em particular pode ser usada em outros lugares em um programa onde uma funcionalidade similar é necessária.

### Grupo Graph e Instances

A funcionalidade total de uma máquina ou planta pode ser descrita pela combinação de diagramas de estados paralelos.

Todos os diagramas de estado são desenhados dentro de um programa S7 são armazenados centralmente na pasta "Sources". A partir daqui você pode inserir e chamá-los sempre que você desejar em um ou mais grupos graph.

Um diagrama de estado chamado em um grupo graph é referenciado como um instance.

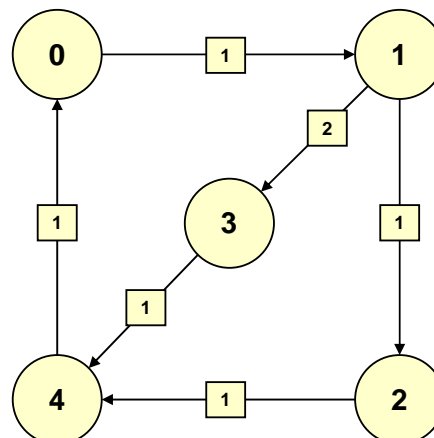
## Elementos de um Diagrama de Estados

### Estados 0,1, ...

- Representados por círculos
- Estados estáticos
- Estados dinâmicos
- Sempre um estado ativo
- Ações são atribuídas aos estados

### Transições

- Representadas por setas
- Condições de transições e ações são atribuídas as transições



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.19



Conhecimento em Automação  
Training Center

### Estados

Um diagrama de estado é um gráfico (graph) contínuo e direto, que representa todos os estados de uma unidade funcional como círculos e todas as transições como setas.

Os estados de uma unidade funcional podem ser do tipo estático (Porta\_aberta, Motor\_desligado, etc.) ou do tipo dinâmico, isto é, estados de movimento (Porta\_abrindo, Motor\_girando\_esquerda, etc.).

A qualquer tempo, o subsistema descrito por um diagrama de estados ou graph, está exatamente em um estado.

### Ações

Ações podem ser atribuídas a estado em diagramas de estados. Estas ações podem ser subdivididas em:

- Ações que são executadas uma vez quando inicia um novo estado.
- Ações que são executadas ciclicamente, tão longa quanto a unidade funcional esteja em seu respectivo estado.
- Ações que são executadas uma vez quando deixando um estado.

As ações são formuladas em uma linguagem similar a STL.

### Transições

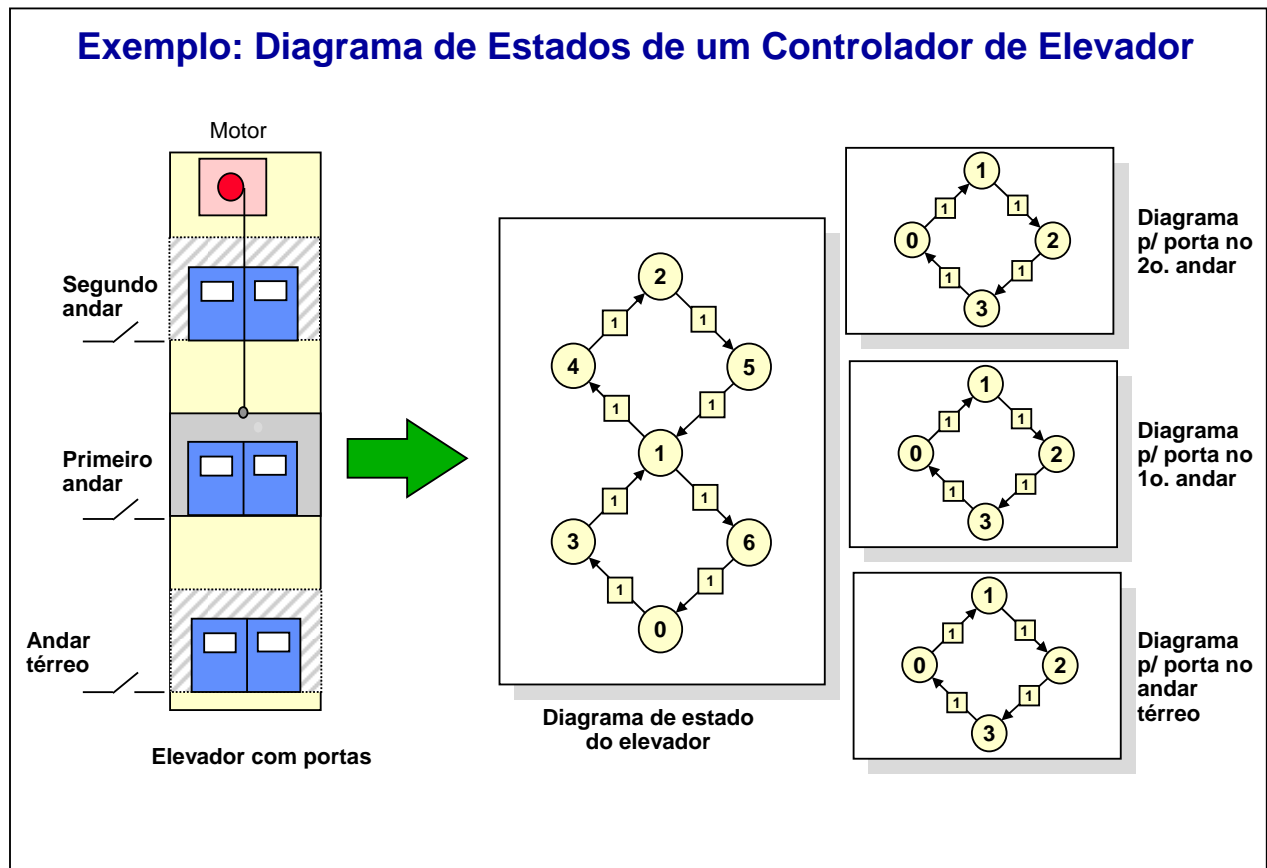
As transições identificam as mudanças de estado de uma unidade funcional. As transições entre os estados são dependentes das condições que são verdadeiras ou não verdadeiras em qualquer instante (tempo).

O subsistema descrito pelo diagrama de estado muda seu estado quando a condição que define sua transição para outro estado for satisfeita. O máximo de uma transição ocorre por ciclo para cada diagrama de estado.

As condições de transição também são formuladas em uma linguagem similar a STL..

Para cada transição uma ação pode ser formulada, a qual será realizada quando a transição ocorrer.

## Exemplo: Diagrama de Estados de um Controlador de Elevador



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.20Conhecimento em Automação  
Training Center

### Vista Geral

O exemplo acima demonstra o método do diagrama de estado por meio de um elevador em um prédio de três andares.

### Separação em FUS

O objeto a ser automatizado (elevador com portas) pode ser separado nas seguintes unidades funcionais mecânicas com relação aos diagramas de estado:

- Cabine do elevador incluindo controle
- Uma porta em cada andar

### Diagrama de estado p/cabine do elevador

Dentro de um diagrama de estado, ou graph, os círculos representam os possíveis estados da unidade funcional e as setas representam as transições de estados.

Para a unidade funcional "cabine do elevador", os estados 0, 1 e 2 representam as posições de parada da cabine do elevador nos respectivos andares.

Os estados 3, 4 e 5 e 6, 7 e 8 representam os movimentos dinâmicos de subida e descida da cabine do elevador entre os andares.

### Diagrama de estado p/ as portas

Na unidade funcional "porta" os estados 0 e 1 representam os estados estáticos "Porta está fechada" e "Porta está aberta", os estados 2 e 3 representam "Porta está abrindo" e "Porta está fechando".

### Mensagens

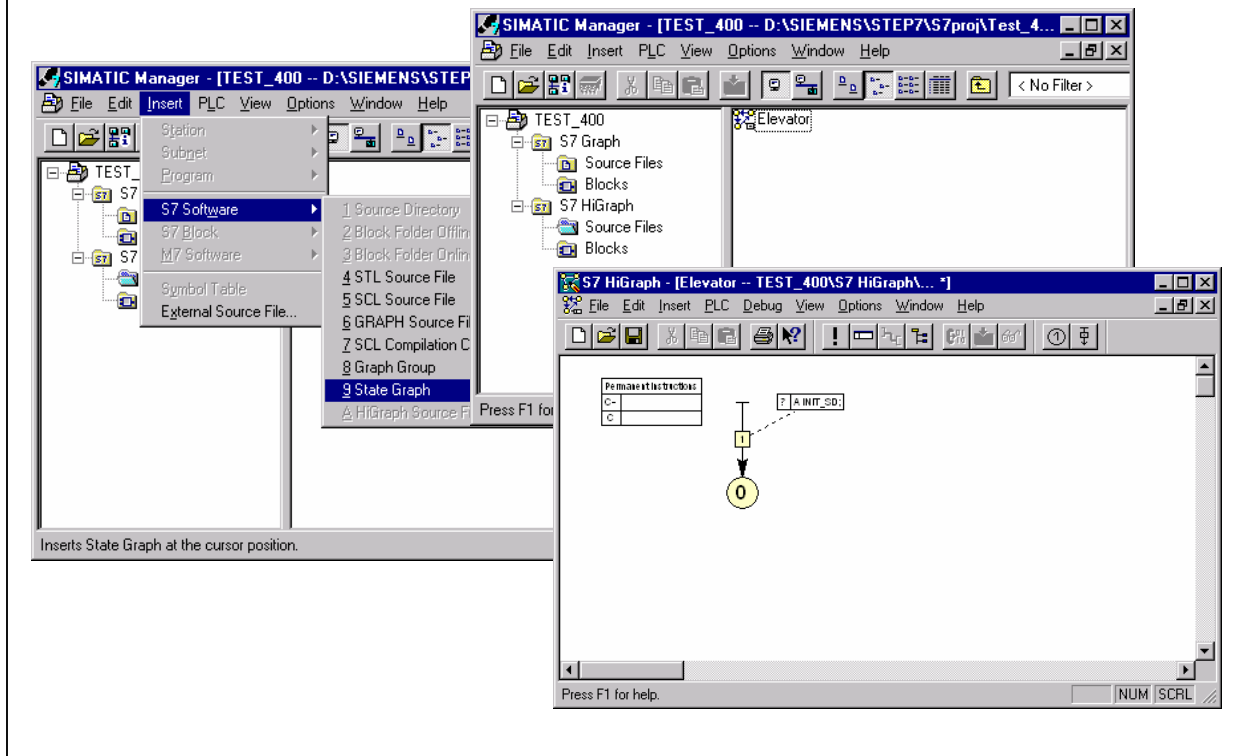
A coordenação do diagrama de estados para a cabine do elevador com os diagramas de estado individuais das portas podem ocorrer sem um diagrama coordenador adicional.

Na busca do "andar desejado" o diagrama de estado "Cabine do Elevador" envia a mensagem "Porta\_aberta" para o diagrama de estado associado "Porta".

Este diagrama de estado recebe a mensagem "abre" a porta, isto é, no recebimento da mensagem a correspondente transição é executada. Quando a porta é fechada novamente, o diagrama de estado "Porta" envia a mensagem "Porta\_fechada" para o diagrama de estado "Cabine do Elevador".

A cabine do elevador pode então mover-se para a próxima porta desejada.

## Criando um Diagrama de Estados



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.21



Conhecimento em Automação  
Training Center

### Vista Geral

De forma a editar diagramas de estado, S7-HiGraph requer um projeto existente. Você deve ter criado primeiro este com o gerenciador SIMATIC antes de você convocar o editor HiGraph.

### Inserindo um Diagrama de estado

Para inserir um diagrama de estado com o gerenciador SIMATIC, proceda como abaixo:

1. Abra a pasta de arquivos fonte (source).
2. Selecione a opção de menu *Insert -> S7 Software -> State Graph*.

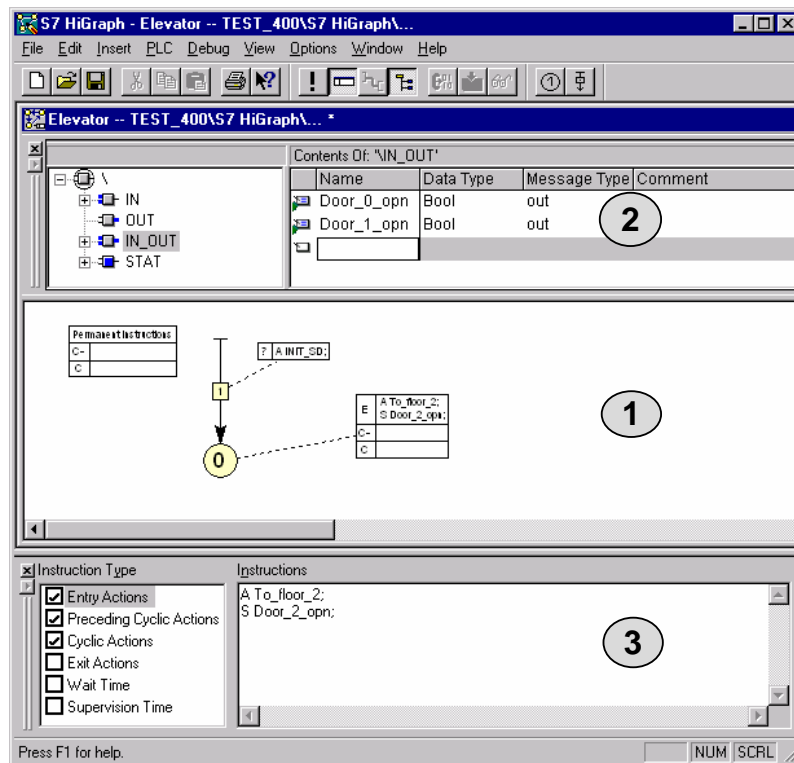
Um diagrama de estados é criado com um nome padrão (default) na pasta *Sources*. Antes de você iniciar a edição, você deve mudar o nome do diagrama de estado (p.ex. Elevador).

3. Duplo clique no diagrama de estado. O editor HiGraph é iniciado e o diagrama de estado é aberto.

### Estado Inicial

O diagrama de estado que você já tinha aberto contém um estado com o número 0 e uma transição ANY. O estado com o número 0 é o estado inicial do diagrama de estados. Este é automaticamente assumido quando o equipamento é ligado (fornecida alimentação).

## A Interface do Usuário HiGraph



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.22



Conhecimento em Automação  
Training Center

### Interface do Usuário

A interface do usuário HiGraph consiste de várias janelas, as quais você pode mostrar ou não, como desejado. Para fazer uso otimizado do espaço disponível da tela, você pode mudar o tamanho da janela e movê-la para onde quiser.

Adicionalmente a janela de edição (1) na qual você edita os diagramas de estado (graphs) e grupos de graph, você também pode usar as seguintes janelas:

- A janela de declaração de variáveis (2) é usada para declaração de variáveis do diagrama de estados (graph). Você mostra esta janela pela seleção da opção de menu *View -> Variables*.
- Você usa a janela de entrada de instruções (3) para programar o conteúdo dos estados, transições e instruções permanentes. Você mostra esta janela pela seleção da opção de menu *View -> Instructions*.
- Existe outra janela na qual erros e mensagens de alerta chegam durante a compilação de um grupo graph são mostradas. Esta janela é automaticamente aberta após cada compilação. Você também pode mostrá-la pela seleção da opção de menu *View -> Messages*.
- A janela de entradas para parâmetros atuais somente está disponível se um grupo graph está aberto. você usa esta janela para atribuição de parâmetros atuais de instances. Você também pode abrir esta janela pela seleção da opção de menu *View -> Actual Values*.

### Declaração de Variáveis

Na declaração de variáveis você declara as variáveis locais e parâmetros dos diagramas de estado. Você também declara as variáveis a serem usadas para trocas de mensagens.

A declaração de variáveis consiste das seguintes seções:

IN: contem os parâmetros de entrada do diagrama de estado e as variáveis pré-definidas "Modo Automático" e "Modo Manual".

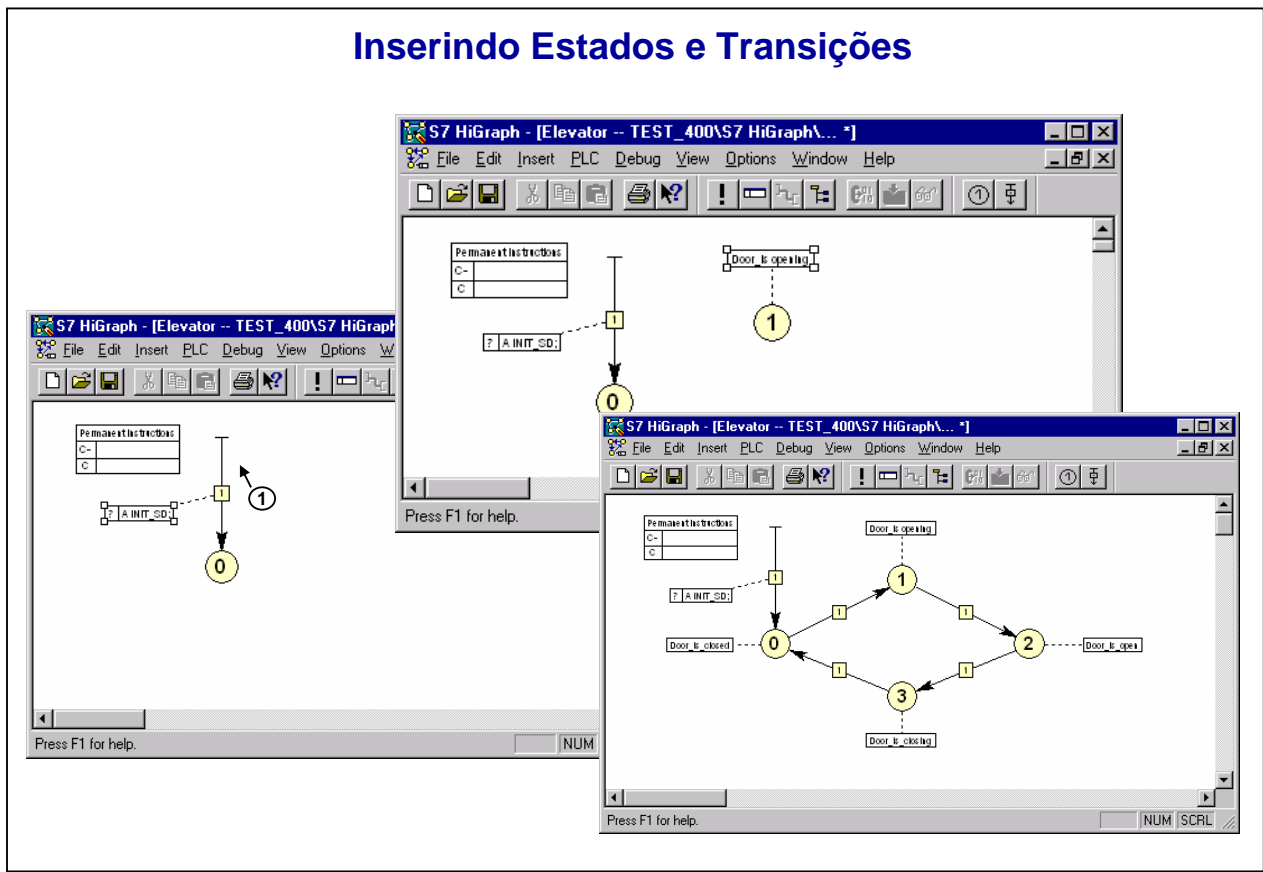
OUT: contem os parâmetros de saída do diagrama de estado.

IN\_OUT: contem os parâmetros de entrada/saída do diagrama de estado e as variáveis usadas para troca de mensagens.

STAT: contem variáveis estáticas e variáveis pré-definidas pelo HiGraph.



## Inserindo Estados e Transições



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.23



Conhecimento em Automação  
Training Center

### Inserindo Estados

Para inserir novos estados dentro de um diagrama de estados (graph), proceda como abaixo:

1. Com uma janela de edição selecionada, escolha a opção de menu *Insert -> State*. O cursor assume a marcação de um círculo com o próximo número de estado livre.
2. Posicione o cursor na posição desejada e clique com o botão do lado esquerdo.

Na posição desejada, um círculo é automaticamente inserido para um estado com o número default 0, 1, 2, etc. Repita o este procedimento até que você tenha inserido todos os estados necessários para a descrição do estado graph.

3. Saia do modo entrada clicando com o em um espaço vazio da janela de edição com o botão esquerdo do mouse.

Cada estado tem um número que é único dentro do diagrama de estado. Para tornar claro o diagrama, você pode dar um nome para cada estado pela seleção da opção de menu *Edit -> Object Properties*.

### Inserindo Transições

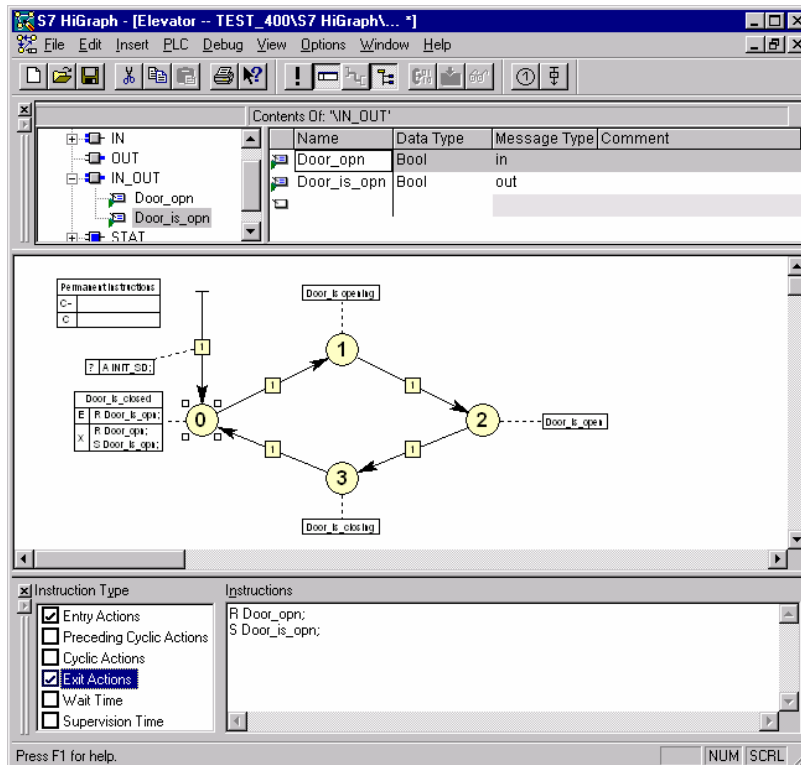
Para inserir transições entre estados individuais, proceda como abaixo:

1. Selecione o item do menu *Insert -> Transition*. O cursor assume a marcação do símbolo de transição.
2. Então clique com o botão esquerdo do mouse no estado inicial e mantenha o botão pressionado, arraste a seta para o estado objetivo.  
Uma transição é inserida entre o estado inicial e o estado objetivo.
3. Saia do modo entrada clicando com o botão esquerdo do mouse.

Do mesmo modo que com os estados, você pode dar nomes para as transições pela seleção da opção de menu *Edit -> Object Properties*.



## Programando Ações



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.24



Conhecimento em Automação  
Training Center

### Tipos de Ações

Você pode programar ações para cada estado na janela de instruções. As ações são subdivididas nos seguintes tipos:

**Entry actions (E):** Ações que são realizadas somente uma vez no início do estado.

**Preceding cyclic actions (C-):** Ações que são executadas enquanto no estado antes que as condições de transição tenham sido verificadas.

**Cyclic actions (C):** Ações que são executadas enquanto no estado após as condições de transição terem sido verificadas.

**Exit actions (X):** Ações que são executadas somente uma vez ao deixar o estado.

Para inserir instruções, selecione o tipo de ação no quadro a esquerda da janela de instruções e então insira as instruções em STL no quadro da direita. As ações são mostradas na janela editor.

### Notas

O RLO é sempre =1 no início da execução de uma tabela de instrução.

Para fazer diagramas de estado reutilizáveis, somente variáveis que tenham sido declaradas na janela de declarações devem ser utilizadas. Parâmetros atuais podem então ser atribuídos para estas variáveis quando inserindo o diagrama de estado em um grupo gráfico.

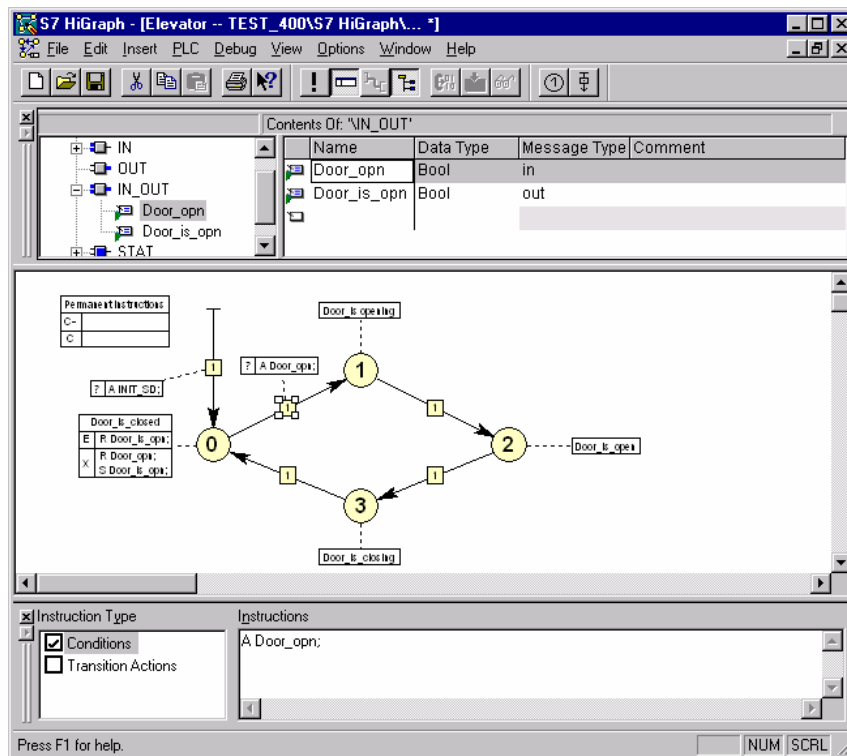
### Tempo de Espera

Você pode especificar se o PLC será mantido em um estado por um intervalo mínimo de tempo antes de verificar as condições de habilitação de transição de passo. Você pode especificar tanto valores constantes como variáveis para o tempo de espera. Você deve então configurar o atributo "Waiting" para as transições que estão com tempo de espera.

### Supervisão de Tempo

A supervisão de tempo é utilizada para monitoração tempo gasto em determinado estado. Se o estado não sair dentro do tempo especificado, a variável pré-definida "ST\_Expired" é setado e uma mensagem de erro é inserida no buffer de diagnóstico.

## Programando Transições



SIMATIC S7

Siemens AG 1999. All rights reserved.

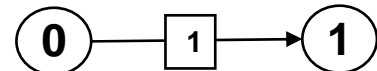
Date: 04.10.2007  
File: PRO2\_13P.25Conhecimento em Automação  
Training Center

### Transições

Uma transição contém as condições para mudança de um estado para outro. Diversas transições de saída podem ser atribuídas a um estado. O HiGraph opera com as seguintes transições:

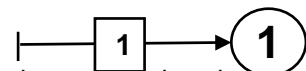
**Transição normal:** Uma transição normal leva de um estado inicial a um estado objetivo.

Ele é representado pelo seguinte símbolo:



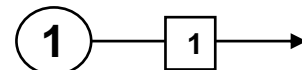
**Transição qualquer:** Uma transição qualquer leva de todos os estados para um estado objetivo. Ele tem alta prioridade sobre todas as outras transições.

Ele é representado pelo seguinte símbolo:



**Transição retorno:** Uma transição retorno leva do estado corrente de volta para o estado que estava previamente ativo.

Ele é representado pelo seguinte símbolo:



### Prioridade

Transições que levam para fora do mesmo estado podem ser arranjadas na ordem certa pela atribuição de diferentes prioridades. Se as condições de transição são satisfeitas ao mesmo tempo, a transição com o nível mais alto de prioridade é ativada.

A mais alta prioridade possível em S7-HiGraph tem o valor numérico 1.

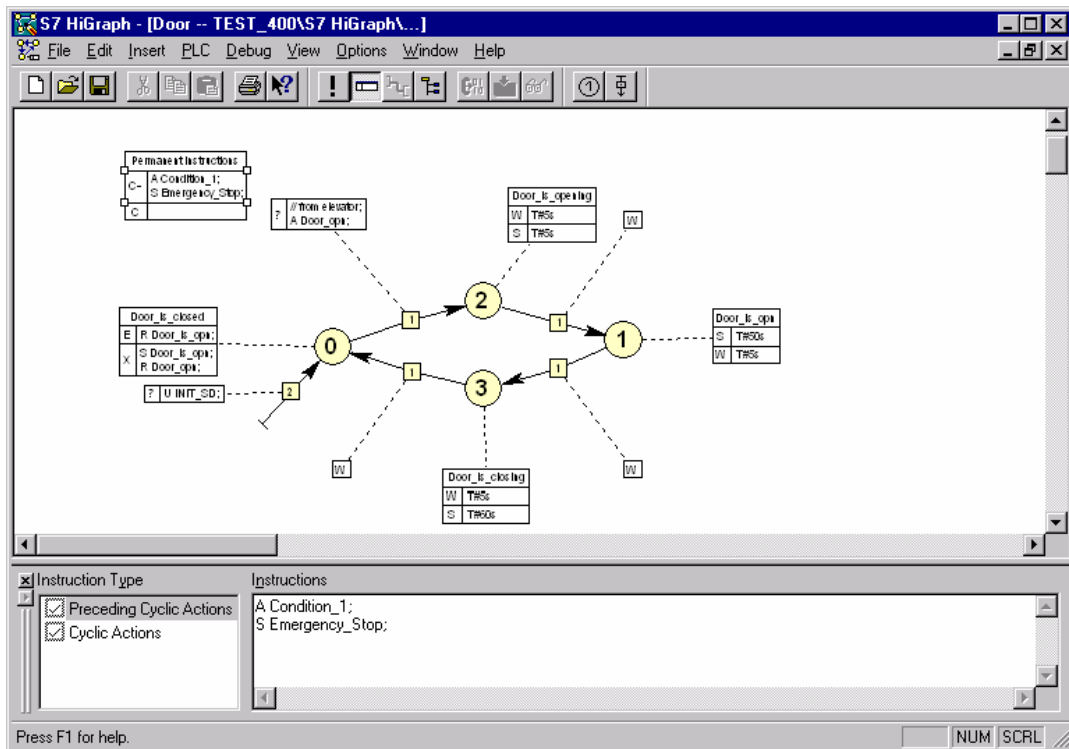
### Instruções

Você pode programar condições e ações para uma transição na janela de instruções:

**Condições (?):** Estas instruções descrevem as condições que devem ser satisfeitas antes de uma mudança de estado possa ocorrer.

**Ações (!):** Estas instruções são executadas uma vez quando a transição é ativada.

## Programando Instruções Permanentes



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.26



Conhecimento em Automação  
Training Center

### Instruções Permanentes

Instruções permanente são executadas uma vez a cada ciclo independentemente do estado corrente. Por exemplo, você pode programar as seguintes atividades centralmente em instruções permanentes:

- Cálculo de variáveis de processo que são verificadas em diversos locais.
- Detecção e processamento de eventos os quais requerem uma resposta não dependente do estado corrente (exemplo: monitoração de uma tela de segurança).

Os seguintes tipos de instruções permanentes estão disponíveis:

Preceding cyclic actions (C-): (ações cíclicas predecessoras) estas são sempre realizadas antes do diagrama de estado atual ser executado.

Subsequent cyclic actions (C): (ações cíclicas subseqüentes) estas são sempre realizadas após o diagrama de estado atual ter sido executado.

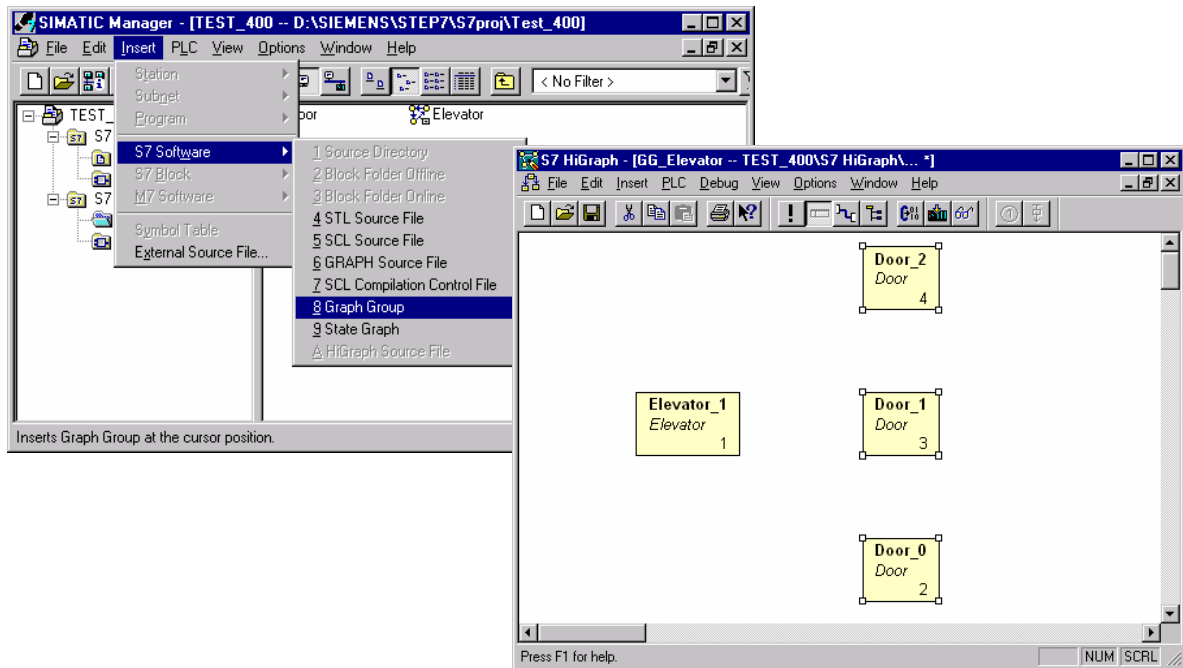
### Programação

Você programa instruções permanentes como segue:

1. Duplo clique na tabela de instruções com o título "Permanent Instructions". A janela de entrada de instruções é aberta.
2. Selecione um tipo de instrução na parte esquerda da janela e insira a instrução em STL na parte direita da janela.

Quando você tiver inserido as instruções, elas serão mostradas na forma de uma tabela na janela de edição para o diagrama de estado.

## Programando Grupos Gráficos (Graph)



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.27



Conhecimento em Automação  
Training Center

### Vista Geral

Os diagramas de estado representam unidades funcionais individuais de uma máquina. De forma a descrever a máquina toda ou planta, os diagramas de estado (graphs) para cada unidade funcional deve ser coletadas juntas em um grupo gráfico.

### Criando um Grupo Gráfico

Para criar um grupo gráfico com o gerenciador SIMATIC, proceda como a seguir:

1. Abra a pasta source na qual você deseja inserir um grupo gráfico.
2. Selecione a opção de menu *Insert -> S7 Software -> Graph Group*.

Um grupo gráfico é criado sobre um nome default (padrão) na pasta *Sources*. Você deve mudar o nome do grupo gráfico antes de você começar a editá-lo (p.ex. Elevador).

3. Duplo clique no grupo gráfico. O editor HiGraph é ativado e um grupo gráfico é aberto.

### “Instanceando” um Diagrama de Estado

A inserção (chamada) de um diagrama de estado (graph) em um grupo gráfico é conhecido como “instanceando”. Quando você tiver inserido o diagrama de estado em um grupo gráfico, você deve atribuir parâmetros atuais para os parâmetros formais declarados nele.

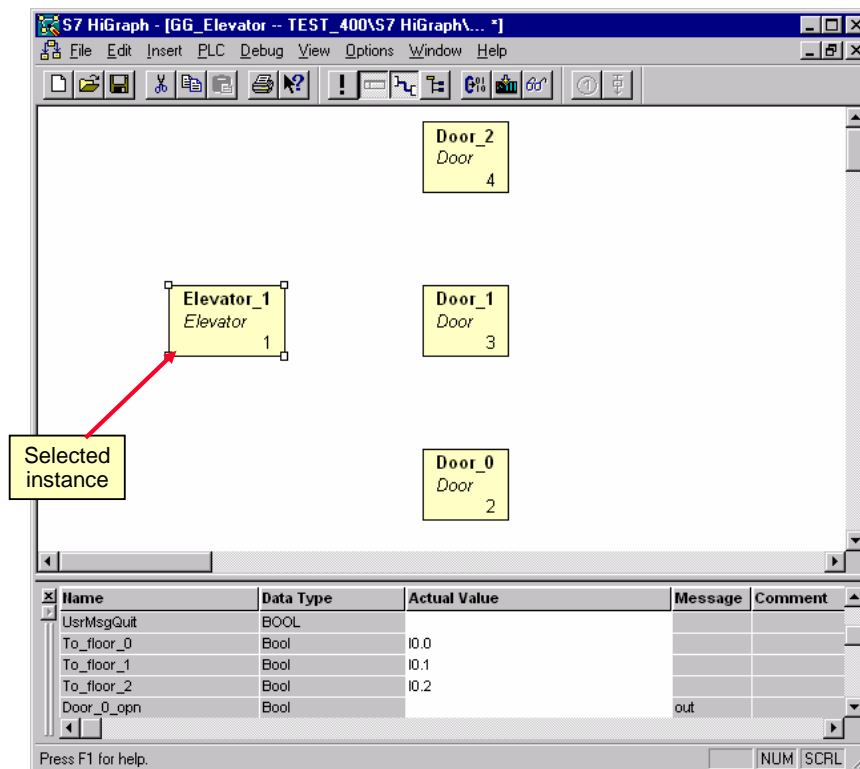
Ele é a atribuição destes parâmetros atuais que habilitam um instance do diagrama de estado a controlar uma unidade funcional real (p.ex. motor, porta de elevador, válvula, etc.). Se diversas unidades funcionais idênticas (p.ex. motores do mesmo tipo, portas de elevadores, etc.) necessitam ser controladas, você pode fazer isto usando diversos instances do mesmo diagrama de estados.

### Inserindo Diagramas de Estado

Para inserir instances de diagramas de estado (graphs) dentro de um grupo gráfico, selecione a opção de menu *Insert -> Instance*.

Você pode então dar aos instances inseridos um nome pela seleção da opção de menu *Edit -> Object Properties*.

## Atribuindo Parâmetros Atuais



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.28



Conhecimento em Automação  
Training Center

### Vista Geral

Para habilitar você a utilizar diagramas de estado repetidas vezes por meio de instanceamento, todos os sinais usados no diagrama de estados devem ser declarados como parâmetros formais na janela de declarações.

Os parâmetros formais salvam espaços para os parâmetros atuais e são atribuídos "sinais atuais" quando um instance é criado.

### Parâmetros Atuais

Quando você tiver inserido os instances de um diagrama de estados em um grupo gráfico, você atribui parâmetros atuais para os parâmetros formais como abaixo:

1. Primeiro selecione o instance requerido do diagrama de estado na janela do grupo gráfico.
2. Selecione a opção de menu *View -> Actual Parameters* para abrir a janela de parâmetros atuais. Os nomes e tipos de dados de todos os parâmetros formais no diagrama de estado aparecem nesta janela.
3. Atribua endereços para os parâmetros formais na coluna "Valores Atuais".

Como parâmetros atuais para um programa HiGraph você pode usar os endereços dos sinais de I/O, memórias bit, contadores, temporizadores, bem como dados e códigos de blocos.

Em seu programa você pode usar endereços absolutos (p.ex. I 1.1, M 2.0, FB 21) ou nomes simbólicos (p.ex. "Motor\_ON").

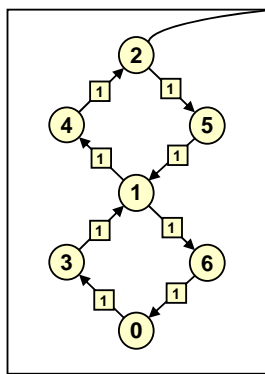
Você usa a tabela de símbolos do seu programa S7 para atribuir nomes simbólicos para endereços absolutos. Você pode trocar entre endereçamento absoluto e simbólico pela seleção da opção de menu *View -> Symbolic Representation*.

### Mensagens?

Você irá encontrar como atribuir mensagens para "parâmetros atuais" nas páginas seguintes.

## Troca de Mensagens entre Diagramas de Estado

Diagrama de estado  
para o elevador



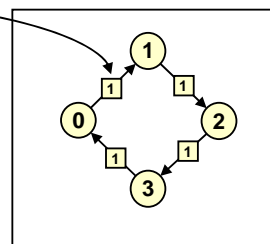
A To\_floor\_2;  
S Door\_2\_opn;

Ação inserida para o estado 2

Name	Data type	Message
Door_2_opn	bool	out

Interface de declaração para elevador

Diagrama de estado  
para a porta



A Door\_opn

Condição de transição p/ t<sub>01</sub>

Name	Data type	Message
Door_opn	bool	in

Interface de declaração para porta

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.29



Conhecimento em Automação  
Training Center

### Vista Geral

Diagramas de estado podem influenciar cada um dos outros pela troca de mensagens. Mensagens são normalmente enviadas por um diagrama fonte (graph) e então avaliada pelo diagrama de destino.

### Mensagens

Uma mensagem é uma variável binária, que pode ser enviada por um graph dentro de sua ação ou parte de transição.

Uma mensagem é sempre enviada para um diagrama de estado (mensagem interna) ou um endereço (mensagem externa).

### Mensagens Internas

Mensagens internas são usadas para sincronização dentro do mesmo grupo gráfico. Elas são mapeadas pelo sistema nos bits do DB associado.

### Mensagens Externas

Mensagens externas são usadas para sincronização de graphs em diferentes grupos gráficos (FCs). Uma variável bit global é declarada quando atribuindo parâmetros atuais.

### Declaração de Mensagens

Você primeiro declara mensagens na seção IN\_OUT da janela de declarações do diagrama de estado (graph). Adicionalmente ao nome e ao tipo de dado (sempre BOOL) da mensagem você também deve especificar o tipo de mensagem, que é uma mensagem de entrada ou saída.

Nome	Tipo de Dado	Mensagem	Comentário
Door_0_opn	bool	out;	//Mensagem de saída
Door_closed	bool	in	//Mensagem de entrada

Você atribui o formato atual da mensagem (mensagem interna) ou a variável bit com a qual a mensagem é conectada (mensagem externa) na janela de parâmetro atual do válido grupo gráfico.

## Atribuindo Valores Atuais para Mensagens

Name	Data Type	Actual Value	Message	Comment
To_floor_1	Bool	I0.1		
To_floor_2	Bool	I0.2		
Door_0_opn	Bool	Door_0.Door_opn	out	
Door_1_opn	Bool	Door_1.Door_opn	out	
Door_2_opn	Bool	Door_2.Door_opn	out	

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.30Conhecimento em Automação  
Training Center

### Vista Geral

Quando você tiver inserido os instances de um diagrama de estado emissor e receptor em um grupo gráfico, você deve informar o instance emissor qual instance está recebendo a mensagem (mensagem interna) ou qual variável bit a mensagem está conectada a (mensagem externa).

### Atribuição de Mensagens Internas

Para atribuir um formato para uma mensagem interna, proceda como abaixo:

1. Selecione o instance do diagrama de estado emissor na janela do grupo gráfico.
2. Na janela de parâmetro atual, selecione a mensagem a ser enviada e insira o instance de recepção na coluna "Valor Atual".

O nome completo do instance de recepção consiste do nome do instance que está recebendo a mensagem e (separado por uma vírgula) o nome da mensagem (type: in), declarada como uma mensagem de entrada no graph de recepção.

Nome	Tipo de Dado	Valor Atual	Mensagem
Door_0_opn	BOOL	Door_0.Door_opn	out
Door_1_opn	BOOL	Door_1.Door_opn	out

Para mensagens de entrada internas você não necessita atribuir um valor atual na janela de parâmetros atuais do instance de recepção.

### Atribuição de Mensagens Externas

Para ligar uma mensagem externa a uma variável bit proceda como abaixo:

1. Selecione o instance do diagrama de estado emissor (graph).
2. Na janela de entrada de parâmetros atuais, selecione a mensagem a ser enviada e insira um endereço de um bit global na coluna "Valor atual".
3. Agora selecione o instance do diagrama de estado de recepção (graph) e atribua o mesmo endereço de bit global a mensagem de entrada válida.



## Salvando e Compilando

### Estabelecendo seqüência de execução

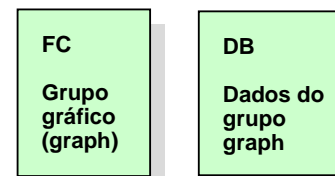
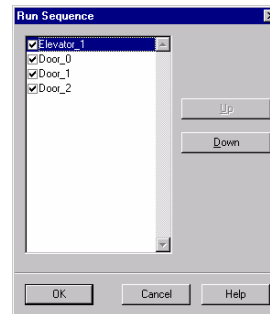
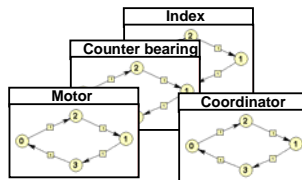
- Menu:  
*Edit -> Execute Order*

### Compilação

- Menu:  
*File -> Compile*

### Integração no OB1

- Atribuição de parâmetro  
INIT\_SD



```
OB1 : Title:
Network 1: Title:
CALL "Elevator"
INIT_SD:=I0.7
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.31



Conhecimento em Automação  
Training Center

### Salvando

Quando você salvar objeto HiGraph, eles são guardados na pasta "Sources" do programa S7 em seu estado corrente. A sintaxe não é verificada. Isto significa que é possível salvar e mais tarde utilizar objetos contendo erros.

Você salva objetos HiGraph pela seleção da opção de menu *File -> Save*.

Por favor notar que qualquer mudança que você fizer em um diagrama de estado afeta todos os instances daqueles que você já tinha inserido em grupos graph.

### Seqüência de Execução dos Códigos

O sistema diagrama de estado é executado ciclicamente. Você pode estabelecer a ordem na qual os instances individuais, dentro de um grupo graph, estão sendo executados pela seleção da opção de menu *Edit -> Execute Order*.

### Compilação

Em HiGraph você somente compila grupos gráficos; você não pode compilar diagramas de estado individualmente. Quando compilando, o HiGraph verifica a sintaxe do programa, gera uma função (FC) e um bloco de dados (DB) e os guarda na pasta "Blocks" do programa S7 válido.

Qualquer erro de sintaxe detectado durante a compilação são suportados na janela de mensagens. Neste caso, nenhum bloco é gerado.

Para compilar um grupo graph, você segue os seguintes passos:

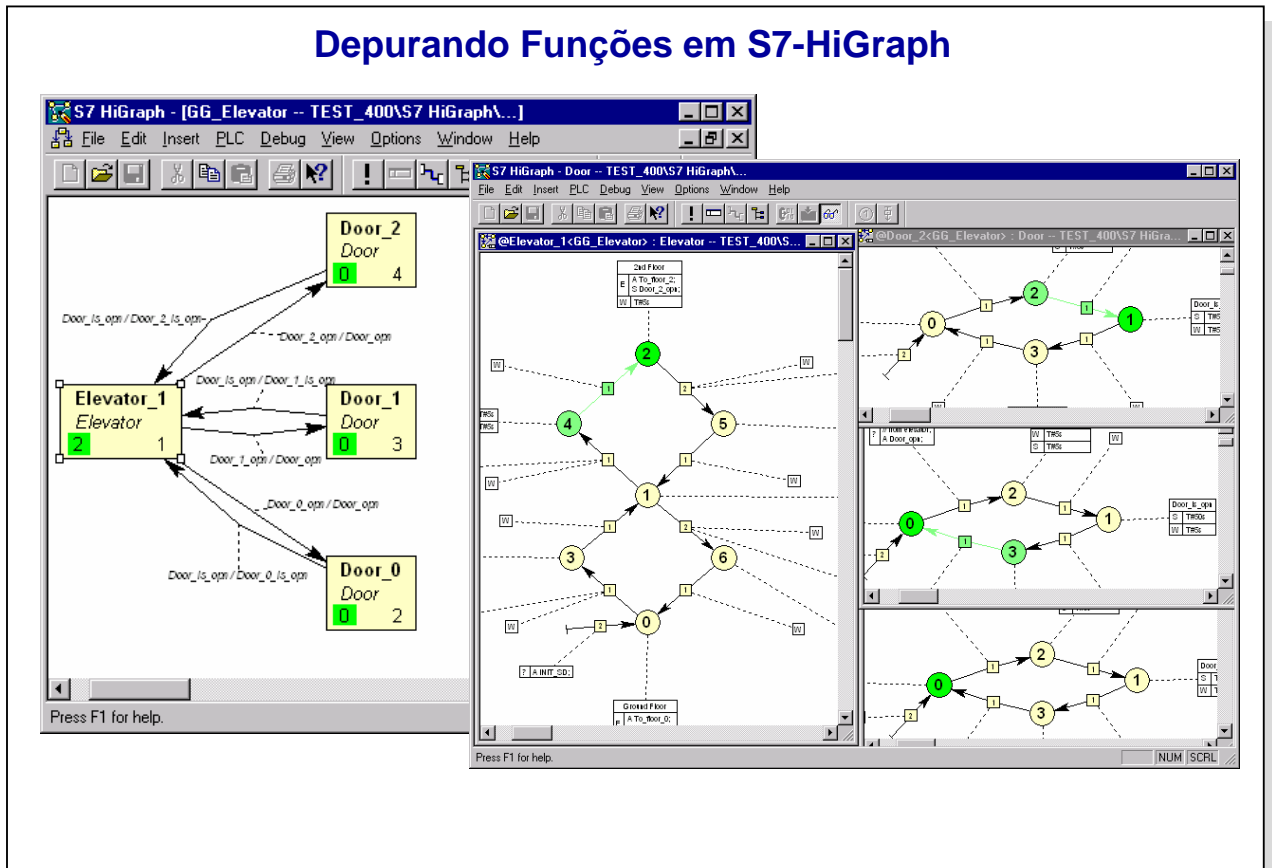
1. Primeiro selecione a opção de menu *Options > Customize* e insere os nomes da FC e do DB, fazendo quaisquer outros ajustes para compilação na página da tabela "Compile".
2. Na janela do grupo graph selecione a opção de menu *File -> Compile*.
3. Procure por qualquer mensagem de erro na janela de mensagens. Para saltar para a posição de erro, simplesmente dê um duplo clique na mensagem de erro.
4. Compile o grupo graph novamente.

### Chamando a FC

Para habilitar o programa HiGraph a ser executado na CPU, a FC deve ser chamada em um bloco que seja executado ciclicamente (p.ex. OB1) e o parâmetro de inicialização INIT\_SD deve ser atribuído.



## Depurando Funções em S7-HiGraph



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.32



Conhecimento em Automação  
Training Center

### Vista Geral

As funções de monitoração habilitam você a monitorar e verificar um programa quando ele está sendo executado na CPU.

O HiGraph fornece as seguintes funções de depuração e monitoração:

- Monitoração do estado lógico do programa
- Monitoração e modificação de variáveis (como em STL/LAD/FBD)
- Avaliação dos dados de referência (como em STL/LAD/FBD)

### Status do Programa

Você pode usar a função de monitoração dos estados lógicos do programa

para verificar a execução de todos os instances em um grupo graph. A execução dos estados e transições individuais é indicado por cores e informações sobre a tabela de instruções também é mostrada.

As seguintes facilidades de monitoração dos estados lógicos do programa são avaliadas em várias janelas HiGraph:

- Janela grupo graph: Aqui você pode visualizar os estados lógicos de todos os instances no grupo graph. O estado corrente é mostrado em cada instance.
- Janela diagrama de estado: Aqui você pode obter informações detalhadas dos estados lógicos de um instance selecionado (estado corrente, transição, etc.).

### Procedimento

Para iniciar a monitoração dos estados lógicos dos programas, proceda como abaixo:

1. Com o grupo graph aberto, selecione a opção de menu *Debug -> Monitor*. A vista geral do status do grupo graph é mostrada.

2. Selecione um ou mais instances e selecione a opção de menu

*Edit -> Open Object* ou dê um duplo clique no instance. Cada instance que você tenha selecionado é aberto on-line e informações detalhadas dos estados lógicos é mostrada.

4. Para monitorar mais instances, mude para vista geral dos estados lógicos e clique no instance que você deseja.

5. Para parar a monitoração dos estados lógicos do programa, desative a opção de menu *Debug -> Monitor*.

## Programando na Linguagem de Alto Nível S7- SCL

### S7-SCL: Linguagem de alto nível para escrita de programas de PLC

- **Compatível com Texto IEC 1131-3 (ST=Structured Text)**
- **Certificado para PLCopen Base Level**
- **Contem todos os elementos típicos de uma linguagem de alto nível, tais como operadores, expressões, instruções de controle**
- **Funções PLC estão integradas (p.ex. acesso I/O, temporizadores, contadores...)**

#### Vantagens:

- **Linguagem estruturada, programas de fácil leitura**
- **Para usuários de linguagens de alto nível**
- **Para algoritmos complexos, grandes quantidades de dados**

#### FUNCTION\_BLOCK Integrator

##### VAR\_INPUT

```
Init   : BOOL;    // Resete valor de saída
x      : REAL;    // Valor de entrada
Ta     : TIME;    // Amostra de tempo em ms
Ti     : TIME;    // Tempo de integração em ms
ulim   : REAL;    // Limite superior do valor de saída
llim   : REAL;    // Limite inferior do valor de saída
```

##### END\_VAR

##### VAR\_OUTPUT

```
y : REAL:= 0.0; // Inicialize valor de saída com 0
```

##### END\_VAR

##### BEGIN

```
IF TIME_TO_DINT(Ti) = 0 THEN // Divisão por ?
```

```
OK := FALSE;
```

```
y := 0.0;
```

```
RETURN;
```

```
END_IF;
```

```
IF Init THEN
```

```
y:= 0.0;
```

```
ELSE
```

```
y := y+TIME_TO_DINT(Ta)*x/TIME_TO_DINT(Ti);
```

```
IF y > ulim THEN y := ulim; END_IF;
```

```
IF y < llim THEN y := llim; END_IF;
```

```
END_IF;
```

```
END_FUNCTION_BLOCK
```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.33



Conhecimento em Automação  
Training Center

### S7-SCL

SCL (Structured Control Language) (Linguagem Estruturada de Controle) é uma linguagem textual de alto nível similar ao PASCAL. Isto simplifica a programação de algoritmos matemáticos e complexas tarefas de processamento de dados para PLCs.

SCL portanto também habilita PLCs S7 a serem utilizados em tarefas mais complexas tais como controle em malha fechada ou avaliação estatística.

Os programas SCL são criados e guardados em uma pasta SCL source (arquivo fonte). Blocos executáveis são então gerados durante a compilação.

SCL é compatível com a linguagem ST (Structured Text) definida na IEC 1131-3 e tem certificação PLCOpen (Base Level).

#### Funcionalidade

SCL oferece um escopo de funcionalidade de uma linguagem de alto nível tal como:

- Loops (malhas)
- Alternativas
- Distribuidor de ramificações, etc.

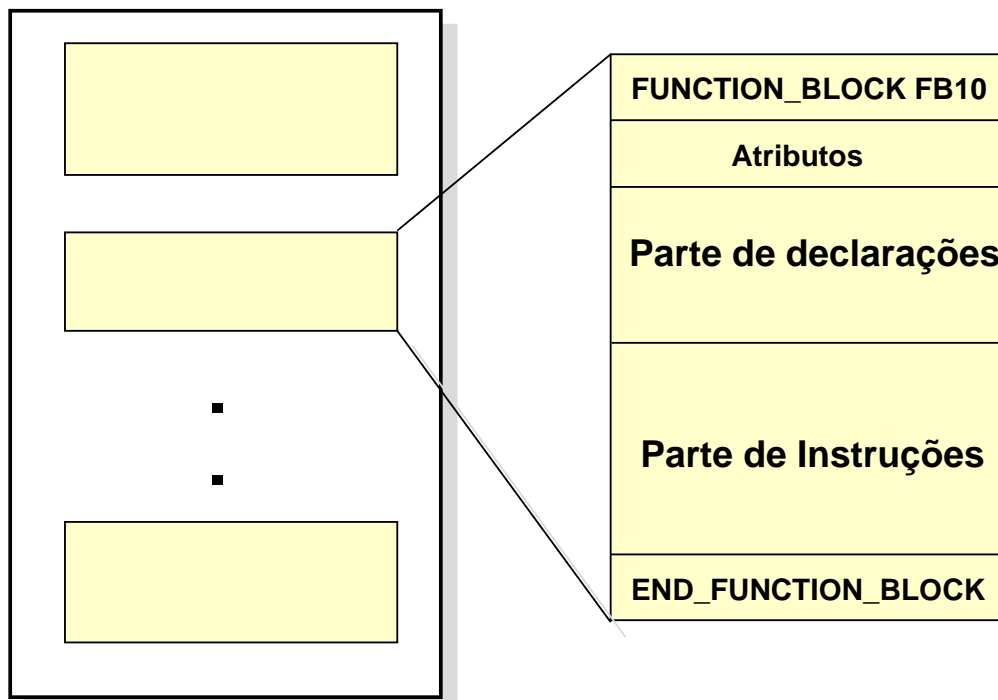
combinado com funções específicas de PLC tais como:

- Acesso binário aos I/Os, memórias bit, temporizadores, contadores etc.
- Acesso a tabela de símbolos
- Acesso a blocos STEP 7

#### Vantagens da SCL

- Linguagem de programação simples de aprender, especialmente para iniciantes
- Simples para escrever programas (inteligíveis).
- Programação simples de algoritmos complexos e processamento de estruturas de dados complexas.
- Depurador integrado para depuração simbólica de códigos fonte (passo simples, breakpoints, etc.).
- Sistema integrado em linguagens S7, tais como STL, LAD e FBD.
- Relativamente fácil para técnicos de PLC para entender através de similaridade com linguagens S7.

## Estrutura de um Arquivo Fonte SCL



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.34



Conhecimento em Automação  
Training Center

### Estrutura de um Arquivo Fonte SCL

Um arquivo fonte SCL pode incluir tantos blocos quanto você deseje (OBs, FBs, FCs, DBs e UDTs).

### Estrutura de um Bloco

Dentro de um arquivo fonte SCL, cada bloco individualmente, dependendo do tipo de bloco, é emoldurado por um identificador padrão para o início e para o fim do bloco. O corpo do bloco é constituído de parte de declaração e parte de instruções.

Como opção, uma parte de atributos pode ser inserida entre o identificador do início do bloco e a parte de declarações.

### Atributos

Atributos identificam propriedades dos blocos, que também podem ser mostradas dentro do gerenciador SIMATIC (Manager) após compilação através do comando *Edit -> Object Properties*.

### Parte de Declaração

As variáveis locais, parâmetros de bloco, constantes e rótulos de saltos são declaradas na parte de declaração de um bloco.

### Parte de Instruções

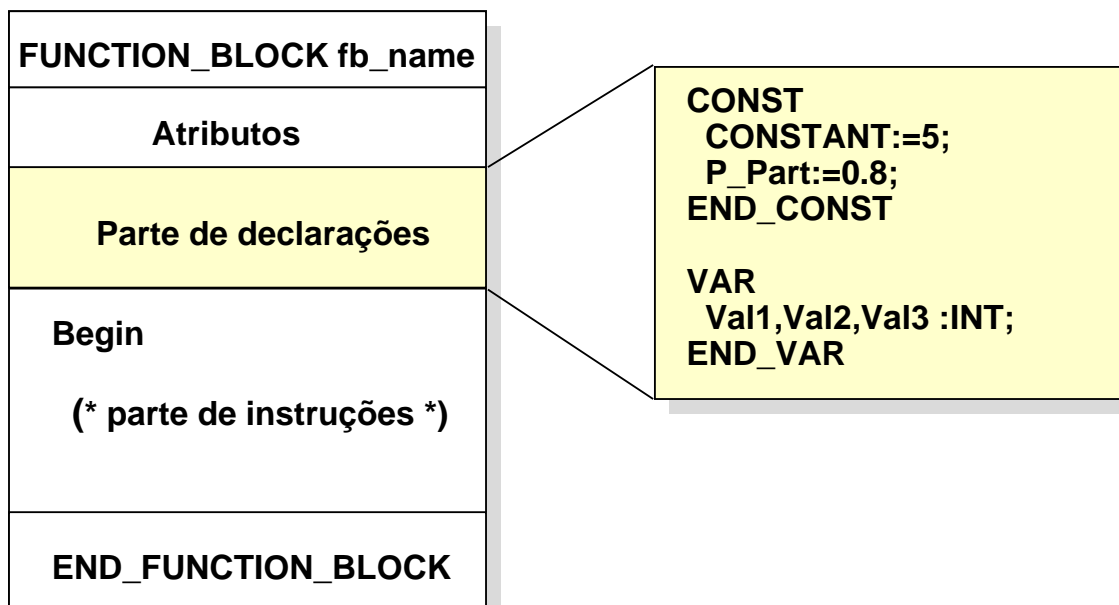
A parte de instruções contem as instruções individuais a serem executadas.

### Seqüência de Bloco

Então para que o seu arquivo fonte SCL possa ser compilado, você deve prestar atenção ao seguinte requisito da seqüência dos blocos:

Blocos chamados devem sempre ser localizados antes da chamada deles.

## A Parte de Declarações de um Bloco



## SIMATIC S7

Siemens AG 1999. All rights reserved.

 Date: 04.10.2007  
 File: PRO2\_13P.35

 Conhecimento em Automação  
 Training Center

### Estrutura

A parte de declarações é usada para definir as variáveis locais e globais, parâmetros do bloco, constantes e rótulos de saltos. Está dividida em blocos de declarações individuais, que estão em cada caso identificadas por seu próprio conjunto de palavras chaves.

Os blocos podem ser inseridos em um arquivo fonte SCL através de *Insert -> Block Template -> Constant, Parameter.*

### Blocos

Dado	Sintaxe	FB	FC	OB	DB	UDT
Constantes	CONST					
	Declaration list	X	X	X		
	END_CONST					
Rótulos de saltos	LABEL					
	Declaration list	X	X	X		
	END_LABEL					
Variáveis Temporárias	VAR_TEMP					
	Declaration list	X	X	X		
	END_VAR					
Variáveis Estáticas	VAR (STRUCT)					
	Declaration list	X			(X)	(X)
	END_VAR					
Parâmetros Entrada	VAR_INPUT					
	Declaration list	X	X			
	END_VAR					
Parâmetros Saída	VAR_OUTPUT					
	Declaration list	X	X			
	END_VAR					
Parâmetros In/Out	VAR_IN_OUT					
	Declaration list	X	X			
	END_VAR					

## A Parte de Instruções do Bloco

FUNCTION\_BLOCK fb\_name

Atributos

Parte de declaração

Begin

(\* parte de instruções \*)

END\_FUNCTION\_BLOCK

Begin

// Exemplo de atribuição de valor  
Measured value:=0 ;

// Exemplo de uma instrução

// de controle

IF I1.1 THEN

N:=0;

SUM:=0.0;

ELSIF START = TRUE THEN

N:=N+1;

SUM:=SUM + IN;

ELSE

OK:=FALSE;

END\_IF;

// Exemplo de chamada subrotina  
FB11.DB22(Pass:=10);

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.36



Conhecimento em Automação  
Training Center

### Parte de Instruções

A parte de instruções contem instruções que são executadas após a chamada do bloco lógico (OB, FB, FC). Estas instruções são usadas para processar dados e endereço ou, no caso de blocos de dados, apresentar valores individuais dentro do DB.

### Sub-divisão

As instruções individuais podem ser essencialmente dividido em três grupos:

- Atribuição de valores: elas são usadas para atribuir uma expressão ou um valor para uma variável.
- Instruções de controle: elas são usadas para criar ramificações dentro do programa ou repetição de grupos de instruções.
- Chamada de sub-rotina: elas são usadas para chamar funções ou blocos de funções.

### Nota

Você deve prestar atenção aos seguintes pontos quando programando instruções:

- A parte das instruções começa com a palavra chave BEGIN e termina com a palavra chave para fim de bloco (p.ex. END\_FUNCTION).
- Cada instrução deve ser fechado com ponto e vírgula.
- Todos os identificadores (nomes) usados na parte de instruções deve ser declarado.

### Templates

Templates para estruturas de controle poden ser inseridas em um arquivo fonte SCL através de *Insert -> Control Structure -> IF, CASE, FOR, WHILE, REPEAT*.

## Expressões, Operadores e Operandos em S7-SCL

### Expressões

- Expressões Matemáticas  $((3+CONST\_INT) * (VAR\_INT ** 37) / 3.14)$
- Expressões de Comparação  $A \geq 9$
- Expressões Lógicas  $(n > 5) \text{ AND } (n < 20)$

### Operadores

- Operador de Atribuição  $:=$
- Operadores Matemáticos  $*, /, \text{MOD}, \text{DIV}, +, -, **$
- Operadores de Comparação  $<, >, \leq, \geq, =, <>$
- Operadores Lógicos NOT, AND or &, XOR, OR

### OPERANDOS

- Constantes 30. 0, FACTOR, 'SIEMENS'
- Variáveis Extendidas Status, IB5, DB10.DW5, Motor.Current, FC12(A:=On)
- Expressões em (...)  $((3+CONST\_INT) * (VAR\_INT ** 37))$

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.37



Conhecimento em Automação  
Training Center

#### Expressões

Expressões consistem de operandos, operadores e parêntesis. Dentro de uma expressão os operadores (p.ex. +, -, \*, /, etc.), isto é, os componentes ativos de uma expressão, são lincadas aos componentes passivos, tais como constantes, variáveis e valores de funções, de modo a formar um novo valor.

Uma expressão portanto calcula o valor que representa.

SCL permite a formação de expressões padrões, isto é, matemáticas, expressões lógicas e comparativas. Variáveis de blocos de dados, arrays, structures e áreas de memória da CPU (entradas, saídas, etc.) podem ser atraídos para a formação.

#### Operadores e Operandos

Expressões consistem de operadores e operandos. Muitos operadores SCL ligam dois operandos (p.ex.  $A + B$ ) e são portanto chamados operadores *binários*. Os outros trabalham somente com um operando e são então chamados operadores *unários*.

O resultado de uma expressão pode

- ser atribuída a uma variável (p.ex.  $A := B + C;$ )
- ser usada como uma condição para instruções de controle (p.ex. IF  $A < B$  DO ... )
- ser usada como um parâmetro atual para a chamada de uma função ou um bloco de funções (p.ex.:  $FB20 \text{ (Input} := A + B \text{)}$ )

## Instruções em S7-SCL

### Valor atribuído

- Exemplo:  $A := B + C;$

### Instruções de Controle

- Instrução IF  $\text{IF E1.1 THEN ... ELSIF ... ELSE ... END\_IF}$
- Instrução CASE  $\text{CASE SELECTOR OF 1: ...; 2: ... ELSE: ... END\_CASE}$
- Instrução FOR  $\text{FOR INDEX := 1 TO 49 BY 2 DO ... END\_FOR}$
- Instrução WHILE  $\text{WHILE INDEX <= 50 DO ... END\_WHILE}$
- Instrução REPEAT  $\text{REPEAT ... UNTIL INDEX:= 51 ... END\_REPEAT}$
- Instrução CONTINUE  $\text{WHILE BOOL\_1 DO ... CONTINUE ... END\_WHILE}$
- Instrução EXIT  $\text{WHILE BOOL\_1 DO ... EXIT ... END\_WHILE}$
- Instrução GOTO  $\text{IF INDEX <23 THEN GOTO MARK; ...}$
- Instrução RETURN  $\text{IF ENABLED THEN RETURN; ...}$

### Chamada de Função e Bloco de Função

- Chamada FB ou SFB  $\text{FB11.DB20(IN:=VAL1, BY:=VAL2);}$
- Chamada FC ou SFC  $\text{RETURN := FC32(IN:=VAL1,OUT:=VAL2);}$

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.38



Conhecimento em Automação  
Training Center

### Instruções

Na parte de declaração de um bloco, as ações são descritas que são para serem executadas com variáveis introduzidas na parte de declaração bem como com dados globais. Normalmente, as instruções são executadas na seqüência na qual elas são listadas em programa texto.

### Atribuição de Valores

Estes são usados para atribuição de novos valores para variáveis. Os valores velhos são então perdidos.

### Instruções de Controle

Estas são usadas para mudança na seqüência na qual as instruções são normalmente processadas.

Uma escolha de várias alternativas na execução em um programa podem ser feitas com instruções condicionais (instruções IF e CASE).

Instruções de Loop (instruções FOR, WHILE e REPEAT) são usados para instruções executadas repetidamente.

Instruções de salto (instruções CONTINUE, EXIT e GOTO) permitem a seqüência de processamento ser interrompida e saltar para um ponto pré-determinado.

### Chamadas FB e FC

De acordo com o princípio de programação estruturada, outras funções (FC e SFC) e blocos de funções também podem ser chamados de um bloco SCL. Blocos que podem ser chamados são:

- funções e blocos de funções adicionais, que foram geradas em SCL ou em outra linguagem STEP 7 (STL, LAD, etc.).
- funções padrões e blocos de funções padrões fornecidas com SCL.
- Funções do sistema (SFC) e blocos de funções do sistema (SFB) que estão disponíveis no sistema operacional da CPU.

## Atribuição de Valores em S7-SCL

### Variáveis Locais

- **Tipos de Dados Elementares** COUNTER := (5 + RUNVAR) \* 2;
- **Estruturas**
  - Estrutura Completa      STRUCT\_1 := STRUCT\_2;
  - Componentes            STRUCT\_1.COMP3 := STRUCT\_2.COMP1;
- **Array**
  - Array Completo        ARRAY\_1 := ARRAY\_2;
  - Componentes        ARRAY\_1[I] := ARRAY\_2 [J];

### Variáveis Globais

- **Área de memória de CPU**
  - Acesso Absoluto      VALUE := IW10;
  - Simbólico            VALUE := INPUT;                      // "Input" na tabela de símbolos
  - Indexado            VALUE := IW[INDEX];
- **Blocos de Dados**
  - Acesso Absoluto      VALUE := DB11.DW5;
  - Simbólico            VALUE := MOTOR.CURRENT;    // MOTOR e CURRENT
  - Indexado            VALUE := MOTOR.DW[Index];    // deve estar na tabela de  
// símbolos
  - Via parâmetros entrada    VALUE := I\_PAR.DW[Index];    // I\_PAR é decl. como VAR\_IN

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.39Conhecimento em Automação  
Training Center

### Princípio

Atribuição de valores recoloca um valor presente de variável com um novo valor, os quais estão especificados em uma expressão. Esta expressão também pode conter identificadores de funções (FC), os quais são deste modo chamadas e retornam valores correspondentes.

O valor de uma expressão atribuída a uma variável deve ser compatível com os tipos de variáveis.

### Atribuição de Valores com Variáveis Complexas

Uma variável complexa representa cada o tipo completo (a estrutura completa, o array completo, a string) ou um componente de variável complexa. Existem então duas possibilidades para atribuição de uma variável complexa. Você pode

- Atribua o conteúdo de outras variáveis complexas completas (struct, array, ou string) para cada variável complexa (structure, array, string).

Por favor notar que, por exemplo, uma estrutura completa somente pode ser atribuída a outra estrutura se os componentes da estrutura coincidirem em seu tipo de dados bem como em seu nome.

Um array completo somente pode ser atribuída a outro array se os tipos de dados dos componentes bem como os limites dos arrays coincidirem exatamente.

- Atribua uma variável tipo compatível, uma expressão tipo compatível ou outro componente para cada componente de uma variável complexa.



## A Instrução IF em S7-SCL

### Sintaxe

```
IF      <expressão> THEN <instruções>
[ELSIF <expressão> THEN <instruções>]    //opcional
.
.
[ELSE  <instruções>]                      //opcional
END_IF
```

### Exemplo

```
IF INPUT_OK THEN
    N := 0;
    SUM := 0.0;
    OK := FALSE;           // Sete o flag OK para FALSE
ELSIF START_OK THEN
    N := N + 1;
    SUM := SUM + IN;
ELSE
    OK := FALSE;
END_IF;
```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.40



Conhecimento em Automação  
Training Center

### Princípio

O problema freqüentemente ocorre em programas, em que várias instruções estão sendo executados dependentes em condições específicas. Isto é possível, com ramificações de programas, ramificar o programa flui por seqüências de instruções alternativas.

A instrução IF é uma instrução condicional. Ela oferece uma ou mais opções e seleciona uma destas instruções para execução (ou nenhuma, se aplicável).

### Execução

A instrução IF é processada de acordo com as seguintes regras:

1. Se o valor da primeira expressão é TRUE, então a parte de instrução após THEN é executada, de outra forma, a expressão na ramificação ELSIF são avaliadas.
2. Se nenhuma expressão booleana é TRUE na ramificação ELSIF, a instrução de seqüência para ELSE é executada (ou nenhuma seqüência de instrução, se o ramo ELSE não existe).

Qualquer número de instruções ELSIF deve existir. Você deve notar que ramificações ELSIF e/ou ramificações ELSE devem ser perdidas. Nestes casos são manipuladas como se estas ramificações existentes com instruções vazias.

### Nota

O uso de uma ou mais ramificações ELSIF como oposição a uma seqüência de instruções IF oferecem a vantagem que as expressões lógicas que seguem uma expressão válida não são a muito avaliadas. O tempo de execução de um programa pode ser então abreviado.

## A Instrução WHILE em S7-SCL

### Sintaxe

```
WHILE <expressão> DO <instruções>  
END_WHILE
```

### Exemplo

```
FUNCTION_BLOCK SEARCH      // SEARCH é declarada na tabela de  
                           // símbolos  
  
VAR  
    INDEX      : INT;  
    KEYWORD    : ARRAY[1..50] OF STRING;  
END_VAR  
  
BEGIN  
    INDEX := 1;  
    WHILE INDEX <= 50 AND KEYWORD[INDEX] <> 'KEY'  
    DO  
        INDEX := INDEX + 2;  
    END_WHILE;  
  
END_FUNCTION_BLOCK
```

SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.41Conhecimento em Automação  
Training Center

### Princípio

A instrução WHILE permite a execução repetida de sequência de instruções na base de uma condição de execução. A condição de execução é formada de acordo com as regras de expressões lógicas.

A parte de instrução que segue o DO é repetida enquanto as condições de execução tem o valor TRUE.

### Execução

A instrução WHILE é processada de acordo com as seguintes regras:

1. A condição de execução é avaliada antes da parte de instruções de cada execução.
2. Se o valor TRUE ocorrer, então a parte de instrução é executado.
3. Se o valor FALSE ocorrer, a execução da instrução WHILE é concluída. Este já pode se o caso na primeira avaliação.

## Chamando Blocos de Funções

### Chamando com DB instance

- Chamada Absoluta  
FB10.DB20(X1 := 5, X2 := IW12, ...); (\* Chamar FB10 com bloco de dados instance DB20 \*)
- Chamada Simbólica  
DRIVE.ON(X1 := 5, X2 := IW12, ...); (\* DRIVE e ON são declarados na tabela de símbolos \*)

### Chamando como um múltiplo instance

- Chamada usando identificador  
VAR  
MOTOR: FB10;  
END\_VAR  
  
BEGIN  
MOTOR(X1 := 5, X2 := IW12, ...); (\* Chamando como um múltiplo instance é somente possível dentro de outros blocos de funções \*)

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.42



Conhecimento em Automação  
Training Center

#### Chamadas de FBs

O conceito múltiplo instance oferecido pelo STEP 7 também está disponível em SCL. Você pode portanto chamar FBs referenciando um bloco de dados instance associado ou usando o modelo múltiplo instance. A chamada de um FB como um múltiplo instance difere da chamada com DBs separados na armazenagem de dados.

No caso de um múltiplo instance, a área de dados necessário não é em um DB separado, mas é encaixado na área de dados instance do chamado FB.

#### Chamada com DB Instance

A chamada é feita em uma instrução especificando:

- o nome do bloco de funções ou bloco de funções do sistema.
- o bloco de dados instance (identificador DB)
- bem como atribuição de parâmetros (parâmetros FB)

Você pode usar cada endereço absoluto ou simbólico na chamada com um DB separado. Você pode chamar FBs com um DB instance separado em todos os blocos lógicos (OB, FB, FC).

#### Chamada como Múltiplo Instance

A chamada é feita em uma instrução especificando:

- o nome local do instance (identificador)
- bem como atribuição de parâmetros (parâmetros FB).

A chamada de um FB como um múltiplo instance somente é possível em FBs. O múltiplo instance também deve ser declarado como uma variável do tipo de dado FBx na parte de declaração (VAR ...END\_VAR) do FB que chama.

## O Flag "OK" para Avaliação de Erro

### Bit Global para detecção de erro (Copiado para o bit BR no fim do bloco)

#### Exemplo:

```

// Setar a variável OK p/ TRUE habilitando
// que uma verificação seja feita a ver se
// as seguintes ações são realizadas
// corretamente

OK := TRUE;
SUM := SUM + IN;
IF OK THEN      // Adição foi realizada corretamente
...
ELSE            // Overflow na adição
...
END_IF;
```

#### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.43



Conhecimento em Automação  
Training Center

#### Descrição

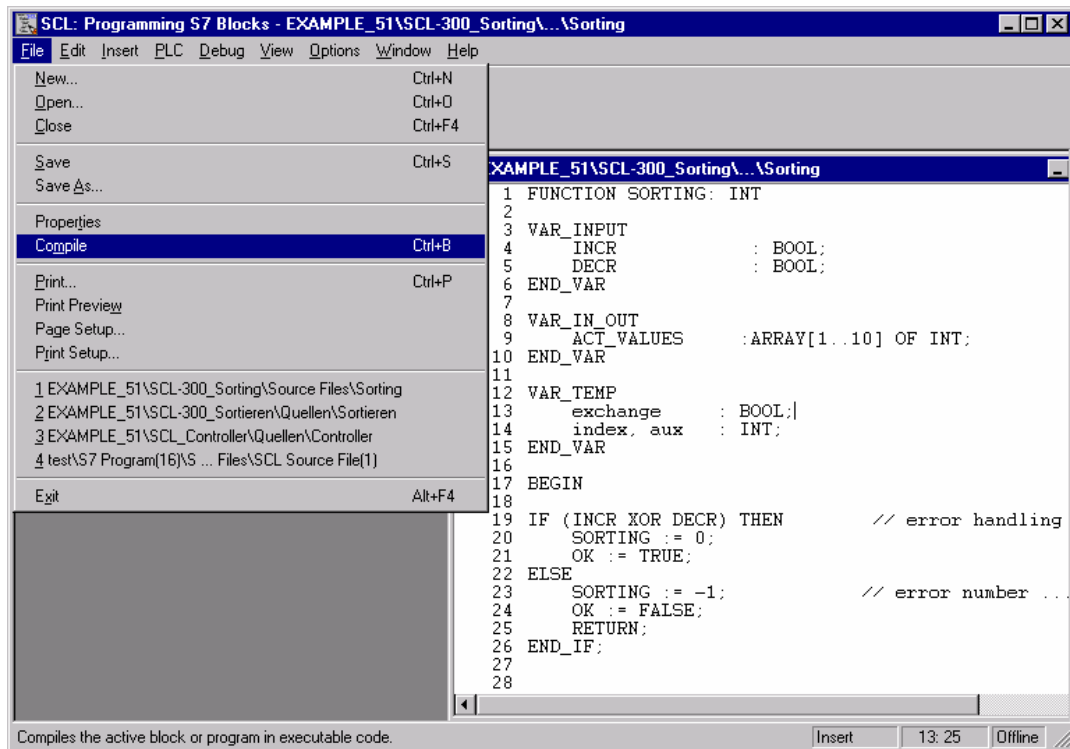
SCL fornece uma variável global do tipo BOOL, conhecido como flag OK, para identificação de erro dentro de blocos lógicos. Este flag é usado para identificar corretamente ou execuções falhas de instruções e para reagir de acordo.

#### Método de Funcionamento

Se um erro ocorre durante a execução de uma instrução (p.ex. overflow), o flag OK esta setado para FALSE pelo sistema. Em cima de um bloco existente, o flag OK é então copiado para o bit BR da palavra de estado da CPU e pode então ser avaliado com o flag OK do bloco chamado.

O flag OK é setado para o valor TRUE no começo da execução do programa. Esta pode ser verificada em qualquer lugar no bloco ou ser setada para TRUE ou FALSE.

## Compilando um Arquivo Fonte SCL



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.44



Conhecimento em Automação  
Training Center

### Compilando

Antes de você poder executar ou testar o seu programa, você deve compila-lo. Você ativa a função de compilação através da opção *Compile* no menu *File* ou através do ícone na barra de ferramentas.

O compilador tem as seguintes propriedades:

- O compilador opera em modo de lote, isto é, ele processa uma fonte SCL como uma unidade. A compilação de blocos individuais em um arquivo fonte SCL não é possível.
- O compilador verifica a sintaxe de um arquivo fonte SCL e sub- seqüentemente mostra todos os erros que ocorrem durante a compilação.
- O compilador cria instruções STL ou informações de teste, se uma fonte SCL está livre de erros e opções correspondentes são setadas. Você deve selecionar a opção *Create Debug Info* para cada programa que você sub-seqüentemente deseja testar em uma linguagem de alto nível.
- O compilador gera para cada bloco de funções chama um bloco de dados instance associated.

### Ajustes

Você adapta a função de compilação através da entrada na opção de menu *Options -> Customize -> Compiler*.

- Create Object Code: Você usa esta opção para especificar se ou não você deseja gerar código executável. Se você ativar a função de compilação sem selecionar esta opção, somente uma verificação de sintaxe ser realizada.
- Optimize Object Code: Cria códigos curtos.
- Monitor Array Limits: Array indexados são checados pela faixa permitida no tempo da execução (runtime). Se um array indexado está for a da faixa permitida, o flag OK é setado para FALSE.
- Create Debug Info: Gera informações de depuração para depurador de linguagem de alto nível.
- Set OK Flag: Esta opção habilita o flag OK a ser checado no arquivo fonte SCL.

## Monitoração Contínua

```

18
19 IF (INCR XOR DECR) THEN          // error handling
20   SORTING := 0;
21   OK := TRUE;
22 ELSE
23   SORTING := -1;                  // error number
24   OK := FALSE;
25   RETURN;
26 END_IF;
27
28
29
30 IF INCR = TRUE THEN
31   REPEAT
32     exchange := FALSE;
33
34     FOR index := 2 TO 10 BY 1 DO
35       IF ACT_VALUES[index-1] > ACT_VALUES[index] THEN
36         aux := ACT_VALUES[index];
37         ACT_VALUES[index] := ACT_VALUES[index-1];
38         ACT_VALUES[index-1] := aux;
39         exchange := TRUE;
40       END_IF;
41     END_FOR;
42   UNTIL NOT exchange
43   END_REPEAT;
44
45   ELSIF DECR = TRUE THEN

```

Press F1 for help. Blockstatus 19: 6 Online

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.45



Conhecimento em Automação  
Training Center

### Seleção

A função Debugger (depuradora) *Monitor Continuously* (monitora continuamente) pode ser selecionada como segue:

1. Tenha certeza que o programa tenha sido compilado com as opções "Create Object Code" e "Debug Information" ativadas.
2. Selecione a janela contendo a fonte do programa a ser testado.
3. Posicione o cursor na linha do texto fonte contendo a primeira instrução da seção que você deseja testar.
4. Selecione a opção de menu *Debug -> Monitor Continuously*.

A área mais larga que pode ser monitorada é determinada e indicada por uma barra cinza no lado esquerdo da janela. Os nomes e valores correntes das variáveis na área sendo monitorada são mostrados no lado direito da janela.

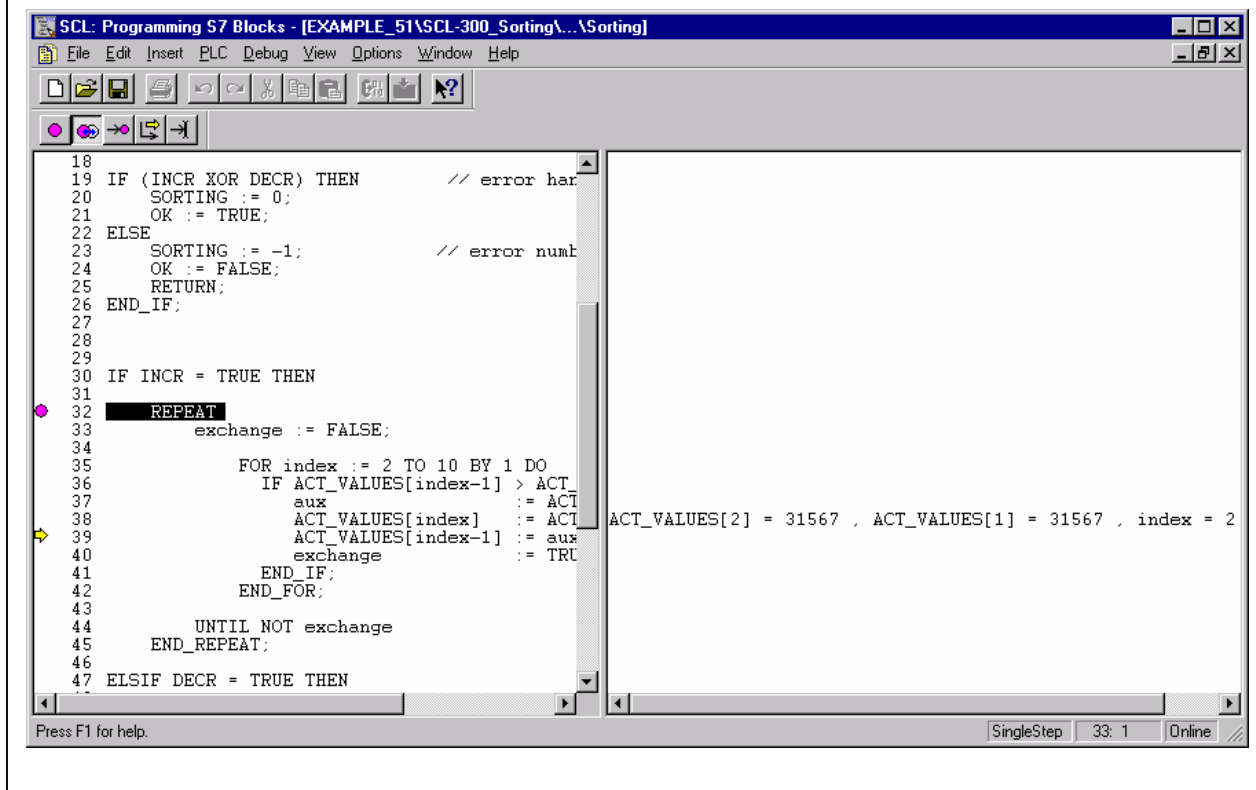
5. Selecione a opção de menu *View -> Symbolic Representation* para ativar ou desativar a monitoração dos nomes simbólicos no programa.
6. Selecione a opção de menu *Debug -> Monitor Continuously* se você deseja interromper a monitoração.
7. Selecione a opção de menu *Debug -> Finish Testing* para parar a monitoração.

### Modo Debug

Você pode mudar a área de monitoração com a entrada *Debug -> Test Environment*.

- Process: neste teste do meio ambiente de processo, o depurador SCL reduz a área máxima de monitoração que então o ciclo de tempo não é ou somente insignificamente prolongado.
- Laboratory: no meio ambiente de Laboratório, a área de monitoração está somente restrita à capacidade da CPU. A máxima área de monitoração é maior do que o meio ambiente de teste de processo.

## Setando e Editando Pontos de Parada (Breakpoints)



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.46



Conhecimento em Automação  
Training Center

### Vista Geral

Quando você selecionar o modo de depuração "Breakpoints Active", você pode monitorar seu programa passo a passo. Você pode executar o programa uma instrução por vez e observar a mudança do conteúdo das variáveis processadas. Após ajuste dos pontos de paradas (breakpoints), você pode executar o programa até um ponto de parada (breakpoint) e monitorá-lo passo a passo de onde estiver.

### Ajustando

#### Breakpoints

Você ajusta breakpoints como abaixo:

1. Abra o fonte SCL do programa que você deseja testar.
2. Defina os pontos de paradas (breakpoints) pelo posicionamento do cursor no local requerido e selecione a opção de menu *Debug -> Set Breakpoint*. Os breakpoints aparecem como círculos vermelhos no canto da janela.
3. Selecione a opção de menu *Debug -> Breakpoints Active*. A janela está verticalmente dividida em duas metades. Quando o próximo breakpoint for encontrado, a CPU vai para o modo HALT, o breakpoint vermelho é marcado com uma seta amarela.
4. Para continuar:
  - Selecione a opção de menu *Debug -> Execute Next Statement*. Quando a CPU tiver processado a próxima instrução, ela irá para o modo HALT novamente.
  - Selecione a opção de menu *Debug -> Continue*. Quando a CPU encontrar o próximo breakpoint, ela irá para o modo HALT novamente.
  - Selecione a opção de menu *Debug -> Execute To Cursor*. Quando a CPU encontrar o ponto selecionado no programa, ela irá para o modo HALT novamente.

Quando o lugar requerido é encontrado, o conteúdo das variáveis correntemente sendo processadas são mostradas no lado direito da janela.

### Número de Breakpoints

O número de breakpoints ativos depende da CPU:

- CPU 416: até 4 breakpoints ativos
- CPU 414: até 2 breakpoints ativos
- CPU 314: um breakpoint ativo

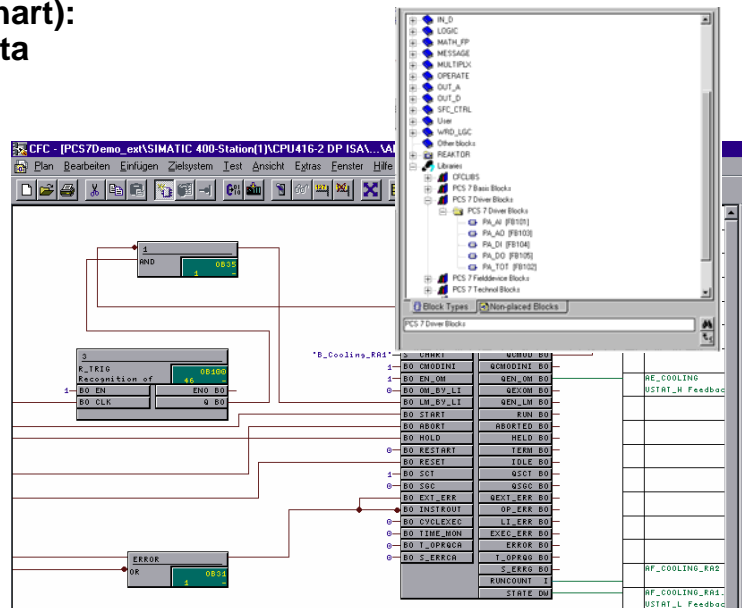
## CFC para SIMATIC S7 e SIMATIC M7

### CFC (Continuous Function Chart): Ferramenta gráfica para escrita de programas PLC

- **Você posiciona os blocos em um folha de desenho e os interconecta**
- **Interconexões são possíveis:**
  - **entre campo I/O**
  - **para blocos em outro gráfico (charts)**
- **Barra margem para gerenciamento de fontes e destinos**

## Vantagens

- **Programação por engenheiros de processos**
- **Alta velocidade p/escrita, depuração e colocação em operação**



## SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2 13P.47



Conhecimento em Automação  
Training Center

## Vista Geral

A ferramenta de engenharia CFC (Continuous Function Chart) habilita você a criar aplicações de automação para SIMATIC S7 ou SIMATIC M7 pelo desenho de uma planta de fluxo de processo - similar ao diagrama de funções para programação de um PLC.

Neste método de programação gráfico, você posiciona os blocos em uma área como uma folha de desenho e interconecta-os graficamente. Com CFC você pode rapidamente e facilmente converter requisitos tecnológicos em programas de automação executáveis.

**Escopo de fornecimento**

O escopo de fornecimento do CFC inclui:

- Editor CFC
- Gerador de código
- Depurador
- Bibliotecas de blocos padrões

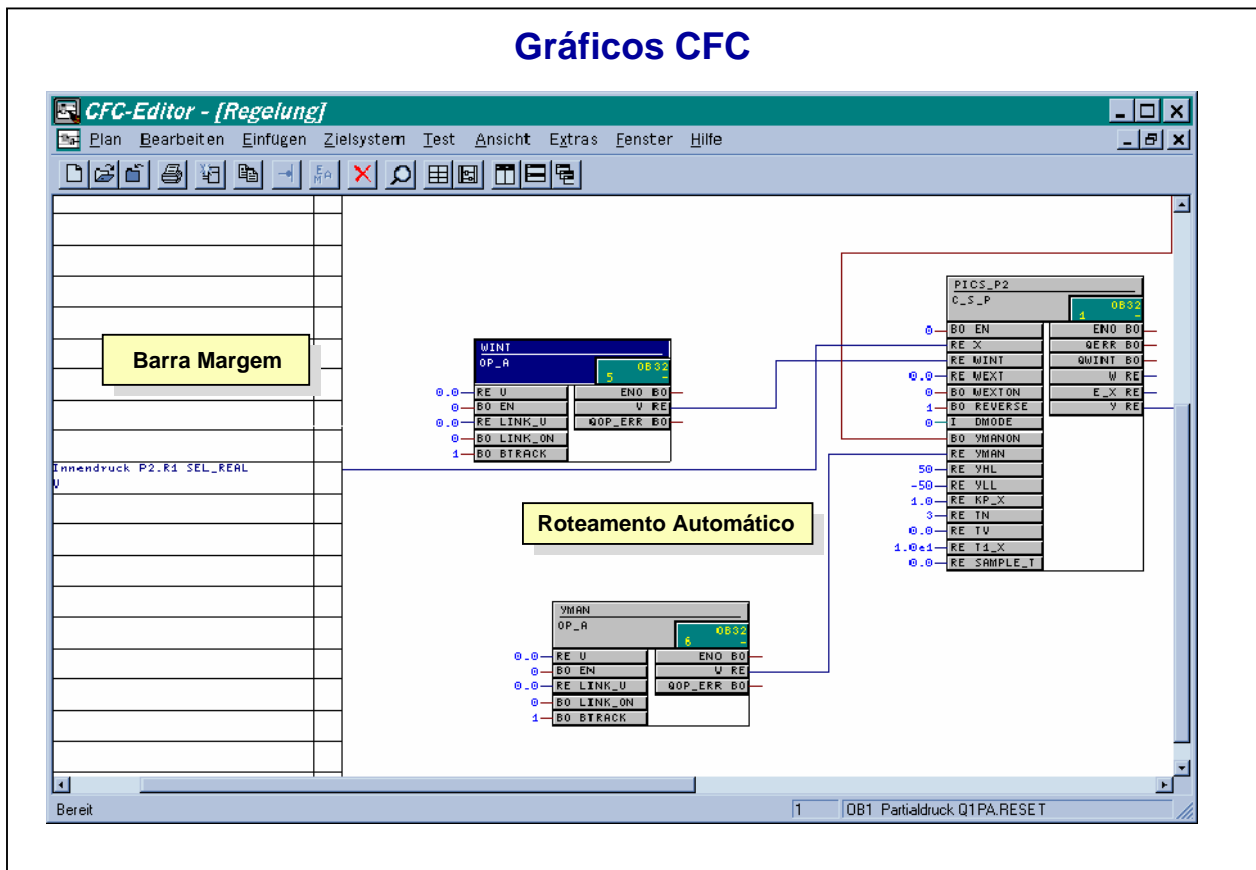
### Benefícios p/ Cliente

O produto CFC pode ser integrado totalmente dentro da arquitetura STEP 7 como um pacote opcional de modo unificado e com gerenciamento universal de dados. Isto torna o CFC fácil de usar e fácil de aprender e fornece dados consistentes.

- CFC pode ser usado para simples tarefas bem como para definição de tarefas muito complexas.
- A tecnologia de simples interconexão torna a comunicação entre blocos amigável ao usuário de configurar.
- Manipulação manual e gerenciamento de recursos de máquina não são muito necessárias.
- Testes amigáveis ao usuário e depuração são oferecidos como suporte.



## Gráficos CFC



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PROZ\_13P.48



Conhecimento em Automação  
Training Center

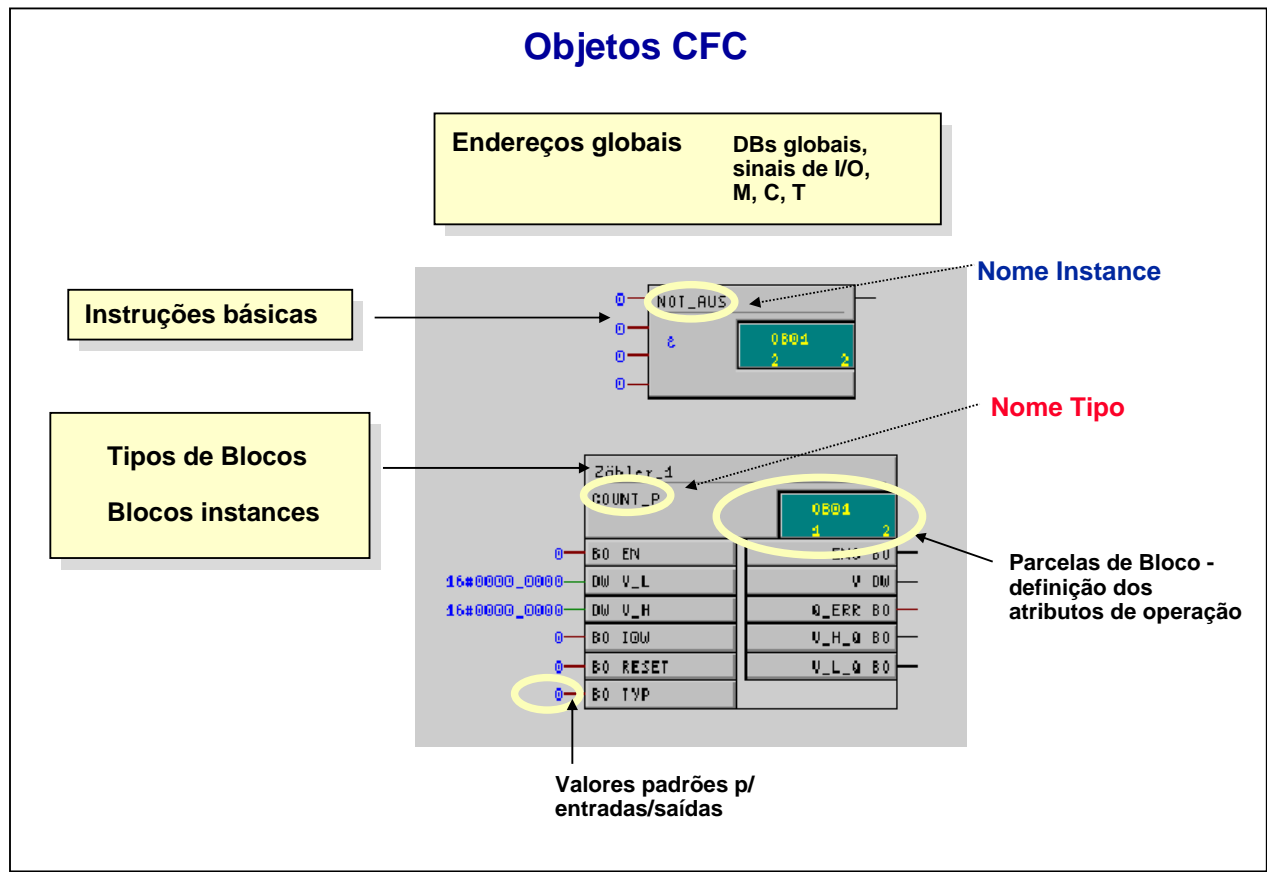
### Gráficos CFC

O bloco instance que você necessita para resolver a definição de uma tarefa tecnológica pode ser dividida em qualquer número de gráficos (charts).

Um gráfico CFC consiste de seis páginas (mostra vista geral)

- 1 página consiste de uma área de trabalho e duas barras margem.
- Automático, gerenciamento da barra marginal de interligação entre charts
- Sinais de monitoração amigáveis ao usuário
- Auto roteadores
- Os recursos são completamente gerenciáveis pelo usuário.
- Documentação 1 para 1 para o conteúdo das informações totais

## Objetos CFC



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.49Conhecimento em Automação  
Training Center

### Objetos CFC

Os mais importantes termos CFC estão assinalados aqui.

### Tipos de Blocos

Um tipo de bloco representa um template (gabarito) para qualquer número de instances e descreve como estes instances estão estruturados internamente. Todos os instances de um tipo de bloco obedecem as mesmas definições básicas como consideradas suas características e sua estrutura de dados.

### Blocos Instances (blocos)

Um bloco instance é um objeto concreto gerado de acordo com seu tipo de descrição. O tipo descrito de características e estrutura de informação para o instance enquanto o estado corrente de cada instance depende de suas operações executadas atualmente e é refletida no conteúdo das informações. Cada instance tem um único identificador que habilita instances a serem distinguidos uns dos outros.

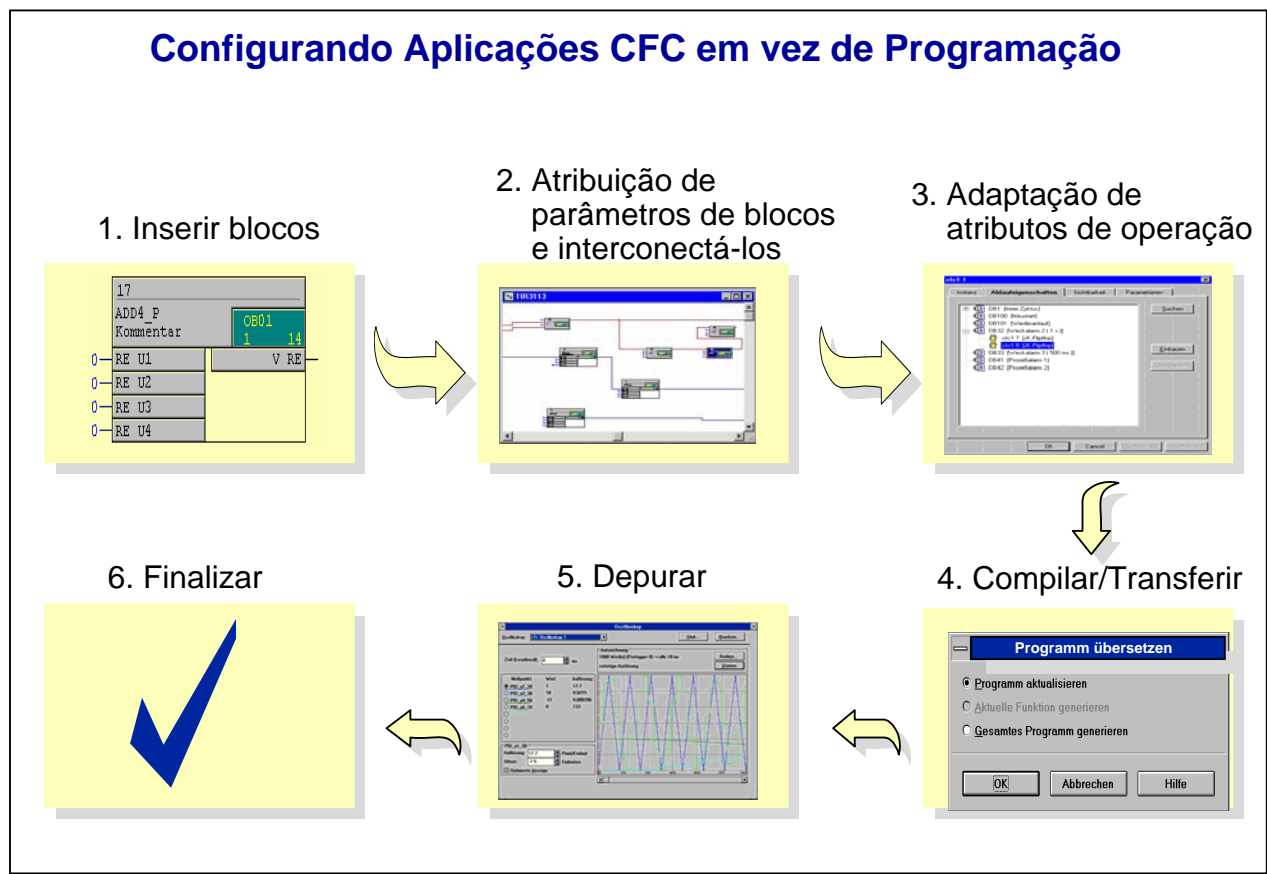
No CFC, o identificador para um bloco instance é feito do nome chart, o qual é único na CPU, o separador '.', e o nome do bloco, o qual é único dentro do chart (máximo de 24 caractere para o nome do bloco).

### Blocos

Na linguagem STEP 7 utilizada, blocos são partes separadas do programa do usuário definido pela sua função, sua estrutura ou seu propósito de aplicação. Existem blocos lógicos (FB, FC,...), blocos de dados e tipos de dados definidos pelo usuário.

- Instruções básicas: funções tais como AND, SUM, etc. contidos no modelo de máquina S7
- Endereços globais: sinais I/O, memórias bit, contadores, temporizadores e blocos de dados globais.

## Configurando Aplicações CFC em vez de Programação



**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.50



Conhecimento em Automação  
Training Center

### Configurando CFC

Você configura uma aplicação CFC como abaixo:

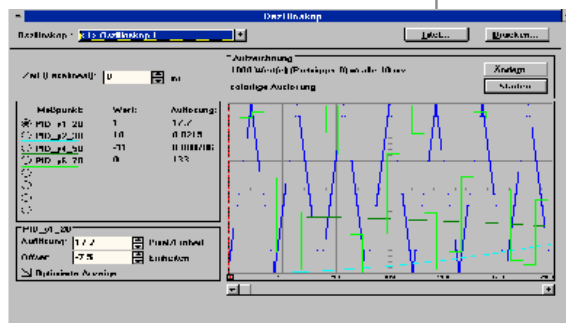
1. Criar um projeto com a pasta charts no programa do usuário S7/M7
2. Criar ou copiar os tipos de blocos
3. Criar um gráfico (chart) CFC
4. Inserir blocos
5. Atribuir parâmetros para os blocos e interconectá-los
6. Adaptação dos atributos de operação dos blocos
7. Compila os gráficos (charts) CFC que você criou
8. Transferir os programas compilados para a CPU
9. Testar os programas transferidos

## Funções Integradas de Teste e Depuração

Variáveis parametrizáveis

Variáveis monitoráveis

Osciloscópio M7



M7: partir, parar e continuar a aplicação, reset, set, Breakpoints, modo passo simples

**SIMATIC S7**

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PROZ\_13P.51



Conhecimento em Automação  
Training Center

**CFC teste e depuração** As seguintes facilidades de teste e depuração estão disponíveis em CFC:

- Monitoração e ajuste de parâmetros para valores de variáveis no chart.
- Controle de fluxo de programa com breakpoints (somente M7)
- Arquivamento de valores variáveis exatos do ciclo (somente M7)

# Configurando Sistemas de Controle Sequencial com S7-SFC

## S7-SFC: Ferramenta para programação de seqüenciadores

- Projetados para requisitos de processos de automação
- Compatível com IEC 1131-3
- Passos de valores atribuídos aos blocos no CFC
- Checa transições das condições de habilitação de passo
- Syntax checked during writing

## Ligue direto com CFC

- Transferência de valores por “Marca&Arrasta”
- Referência cruzada

## Visualização em WinCC

The screenshot displays the S7-SFC Editor interface. The main window shows a ladder logic diagram with the following steps and transitions:

- Step 1: AUFRESEN (Start)
- Transition 1: DRUCK OK
- Step 2: ZULUFT ZU
- Transition 2: V1 ZU
- Step 3: ENTSPANNEN
- Transition 3: MINDRUCK
- Step 4: ABLUFT ZU
- Transition 4: O2 OK
- Step 5: ENDE (End)

Two configuration windows are open:

- STEP: Kühlen** (Step Configuration):

First Operand	Operator	Second Operand
Motor 2	start	
Heizventil 3	open	
Temperat. Sollwert	=	243
Füllicht. Sollwert	=	Rezept 1. Sollwert

- TRANSITION 2** (Transition Configuration):

Condition	Operator	Second Operand
NOT		
V1	initial	
Motor 4	initial	
Temperat. Sollwert	>	125
Motor 4 Threshold	>	71%
Signal 6	=	Rezept 1. Sollwert

**SIMATIC S7**  
Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2 13P.52



Conhecimento em Automação  
Training Center

## SFC (Sequential Function Chart)

SFC é um sistema de controle seqüencial que sempre opera passo a passo, a qual foi especialmente projetada segundo requisitos de automação de processo (engenharia de processos, controle de processos, etc.).

O típico campo de aplicação dos sistemas de controle seqüencial deste tipo são plantas com operação descontínua. Sistemas de controle seqüencial também pode deste modo ser usado em plantas que operem continuamente, por exemplo, para colocação em operação (startup) ou shutdown, mudanças do ponto de trabalho bem como mudanças de estado tipo distúrbios etc.

Com SFC, por exemplo, especificação de manufatura de produtos podem ser escritos como evento dirigido a processo.

## Princípio de Operação

No editor SFC você gera o fluxograma pelo significado gráfico. Os elementos de estrutura gráfica estão posicionados de acordo com regras fixas. Você não deve se preocupar com detalhes tais como algoritmos ou alocação de recursos de máquinas, mas ao contrário podem concentrar em aspectos tecnológicos da configuração.

Após geração de topologia gráfica, você chaveia para vista de detalhes (zoom) e aqueles parâmetros atribuídos a elementos individual, isto é, você configura as ações (passos) e condições (transições).

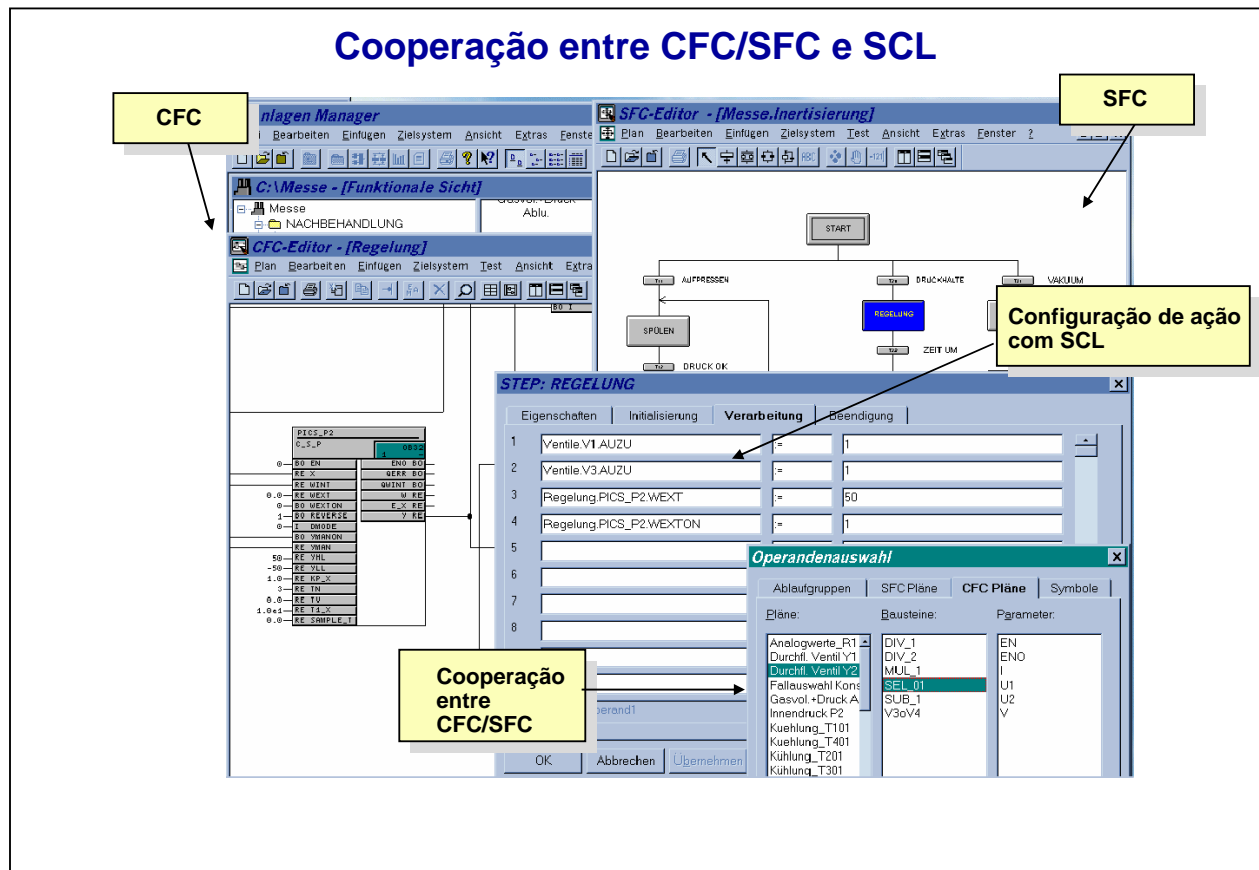
Você normalmente programa ações pela seletiva edição básica de funções de automação criadas com CFC utilizando mudanças de modo e estados.

Após configurar, gerar códigos de máquina executáveis com SFC, transferir para o PLC e testá-la com funções de depuração SFC.

## Volume de Dados de Projeto

- Seqüenciadores por chart 1
- Passos por chart 2 ... 255
- Transições por chart 1 ... 255
- Instruções por passo <= 50
- Condições por transição <= 10

## Cooperação entre CFC/SFC e SCL



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_13P.53

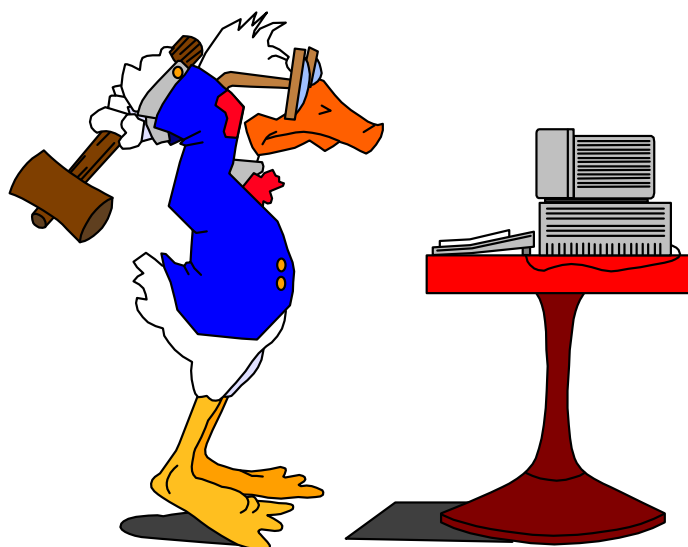


Conhecimento em Automação  
Training Center

### Cooperação

- Programação de blocos em linguagem de alto nível SCL
- Gerenciamento de dados comuns e geração de código com CFC
- Acesso cruzado direto de SFC para blocos instances CFC
- Integração com o gerenciador SIMATIC STEP 7

## Soluções dos Exercícios



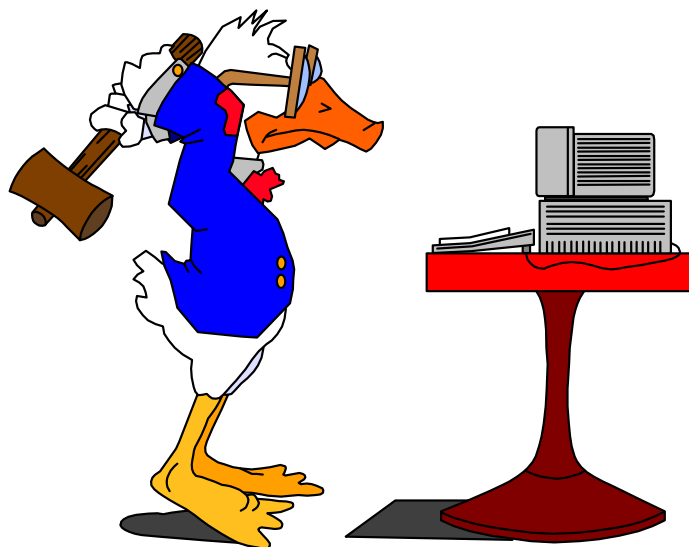
### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.1Conhecimento em Automação  
Training Center

Conteúdo	Pág.
Suporte da Área de Treinamento com S7-300 .....	3
Configuração da Unidade de Treinamento com S7-300 .....	4
Configuração da Unidade de Treinamento com S7-400 .....	5
O Simulador .....	6
O Modelo Correia Transportadora .....	7
Solução do Exercício 1.1: Salto após uma Subtração .....	8
Solução do Exercício 1.2: Salto após uma Multiplicação .....	9
Solução do Exercício 1.3: Programando um Distribuidor de Saltos .....	10
Solução do Exercício 2.1: Cálculo de Expoentes .....	11
Solução do Exercício 2.2: Troca de Dados no ACCU1 .....	12
Solução do Exercício 2.3: Formação de Complementos .....	13
Solução do Exercício 3.1: Calculando a Distância .....	14
Solução do Exercício 4.1: Programação de Loop com Endereçamento Indireto de Memória .....	15
Solução do Exercício 4.2: Programação de Loop com Endereçamento Indireto de Registro .....	17
Solução do Exercício 4.3: Cálculo de Soma e Valor Médio .....	18
Solução do Exercício 5.2: Acessando Tipos de Dados Complexos .....	19
Solução do Exercício 5.3: Leitura do Relógio do Sistema .....	20
Solução do Exercício 6.1a: Planta de Engarrafamento – Modo de Operação .....	21
Solução do Exercício 6.1b: Planta de Engarrafamento – Controle do Transportador .....	22
Solução do Exercício 6.2a: FB1 para Estação de Trabalho .....	26
Solução do Exercício 6.2a: FB2 para o Transportador .....	28
Solução do Exercício 6.2a: OB1.....	30
Solução do Exercício 6.2b: Extensão para 3 Estações .....	31

## Soluções dos Exercícios



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.2



Conhecimento em Automação  
Training Center

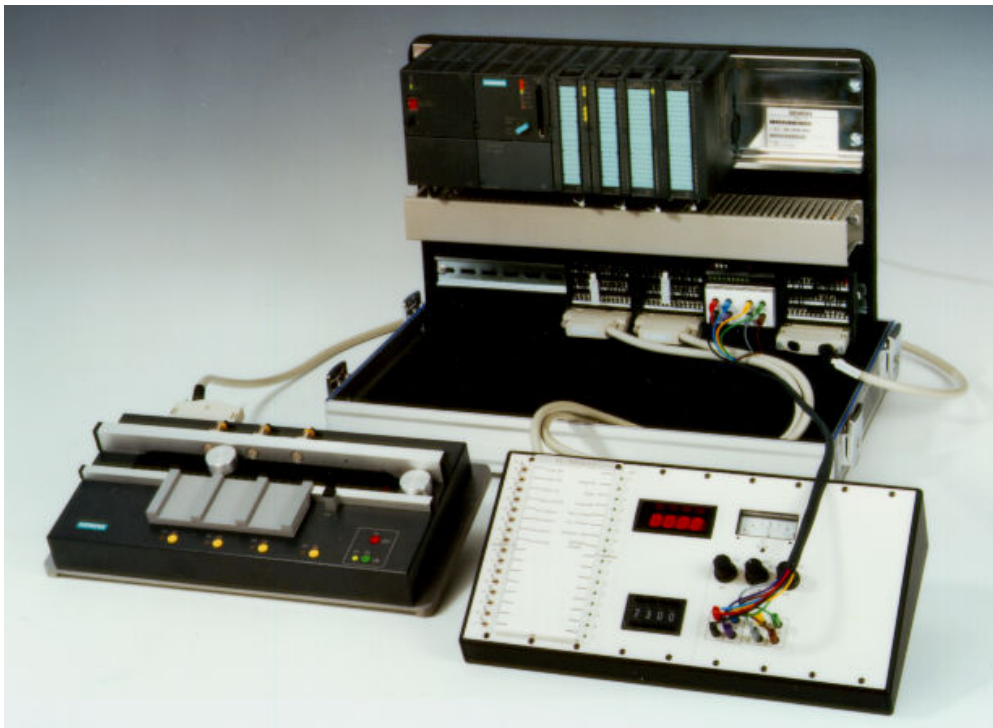
### Conteúdo

### Pág.

Solução do Exercício 7.2: Testando o Bloco de Dados (SFC 24: somente S7 400) .....	35
Solução do Exercício 7.3: Criando um DB (SFC 22) .....	36
Solução do Exercício 7.4: Copiando DB da Memória de Carga para de Trabalho (SFC 20) .....	37
Solução do Exercício 7.5: Inicializando um DB com "0" (SFC 21: FILL) .....	38
Solução do Exercício 7.6: Escrevendo uma Mensagem no Buffer de Diagnóstico (SFC 52) .....	39
Solução do Exercício 7.7: Bloco Contador com "Debouncing de Contato" .....	40
Solução do Exercício 8.1: Manipulação de Erro no FC43 .....	41
Solução do Exercício 9.2: Contagem de Peças Terminadas .....	43
Solução do Exercício 10.2: Comunicação com os SFBs GET/PUT .....	49
Solução do Exercício 10.3: Comunicação com os SFBs START/STOP .....	51



## Suporte da Área de Treinamento com S7-300



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.3



Conhecimento em Automação  
Training Center

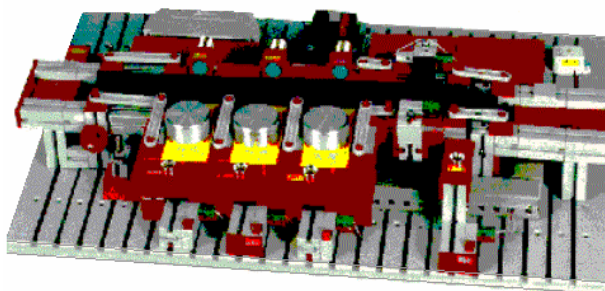
### Conteúdo do Kit de Treinamento

O kit de treinamento consiste dos seguintes componentes:

- Um controlador lógico programável S7-300 com CPU 314 ou CPU 315-DP
- Módulos de entrada e saída digital, módulo analógico
- Simulador com equipamentos para testes digitais e analógicos
- Modelo de Correia Transportadora

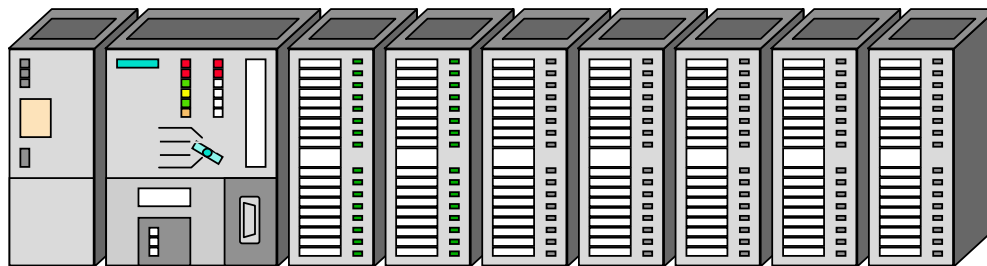
#### Nota:

É bem possível que o seu kit de treinamento de área não seja equipado com o modelo de correia transportadora mostrado acima, mas com o modelo mostrado na foto abaixo.



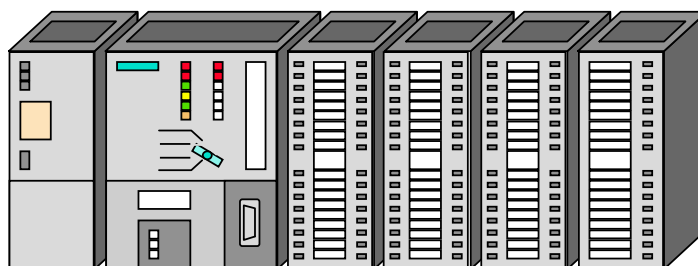
## Configuração da Unidade de Treinamento com S7-300

### Versão A (módulos de 16 canais de I/O)



Módulo	-->	PS	CPU	DI 16	DI 16	DO 16	DO 16	DI 16	DO 16	AI/AO4
No. do Slot	-->	1	2	4	5	6	7	8	9	10
Endereço I/O	-->			0	4	8	12	16	20	352

### Versão B (módulos de 32 canais de I/O)



Módulo	-->	PS	CPU	DI 32	DO 32	DI8/DO8	AI 2
No. do Slot	-->	1	2	4	5	6	7
Endereço I/O	-->			0	4	8	304

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.4



Conhecimento em Automação  
Training Center

### Configuração da Versão A

O controlador programável é configurado com os seguintes módulos:

Slot 1:	Fonte de alimentação 24V/5A	
Slot 2:	CPU 314 ou CPU 315-2 DP	
Slot 4:	Entrada digital 16x24V	Entradas do simulador
Slot 5:	Entrada digital 16x24V	Botões Pushwheel
Slot 6:	Saída digital 16x24V 0.5A	Saídas do simulador
Slot 7:	Saída digital 16x24V 0.5A	Display digital
Slot 8:	Entrada digital 16x24V	Entradas do modelo C. Transp.
Slot 9:	Saída digital 16x24V 0.5A	Saídas do modelo C. Transp.
Slot 10:	Módulo analógico 4 AI/4 AO	Ajustável do simulador

### Configuração da Versão B

O controlador programável é configurado com os seguintes módulos :

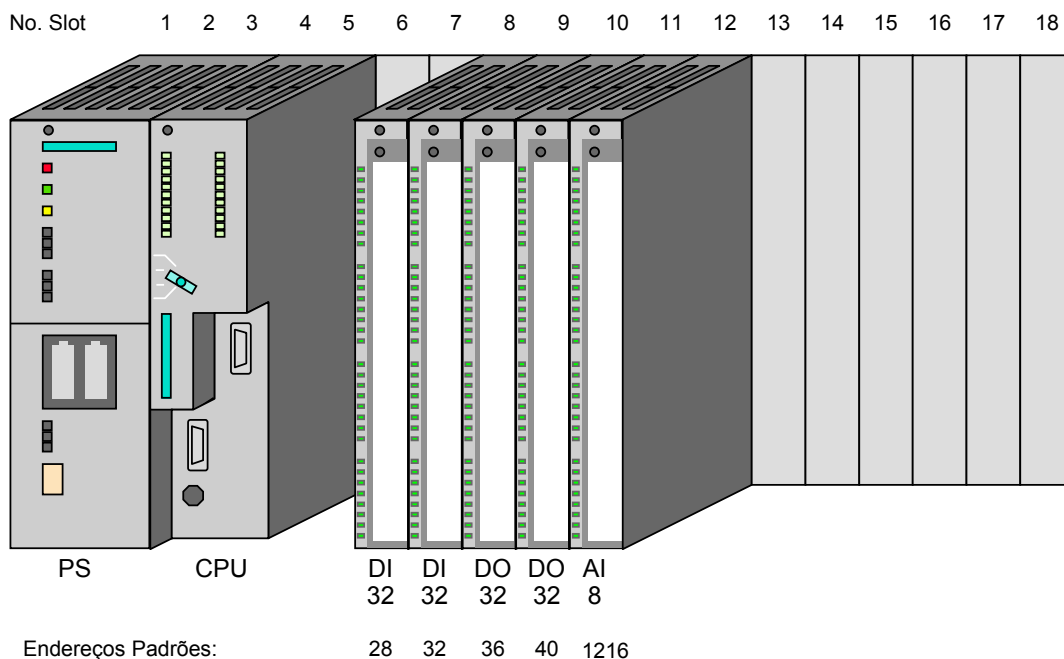
Slot 1:	Fonte de alimentação 24V/5A	
Slot 2:	CPU 314 ou CPU 315-2 DP	
Slot 4:	Entrada digital 32x24V	Entradas do simulador e Botões Pushwheel
Slot 5:	Saída digital 32x24V/0.5A	Saídas do simulador e Display digital
Slot 6:	Entrada e saída digital módulo 8x24V/ 8x24V 0.5A	Modelo C. Transp.
Slot 7:	Entrada analógica 2 AI	Seção analógica do simulador

### Endereços

O endereçamento slot fixo é usado no S7-300 (CPU 312-314). Os endereços do módulos são mostrados no slide.

Os endereços iniciais do módulos podem ser ajustados pela atribuição de parâmetros na CPU 315-2DP e para o S7-400.

## Configuração da Unidade de Treinamento com S7-400



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.5Conhecimento em Automação  
Training Center

### Arquitetura

Você pode ver a arquitetura da unidade de treinamento do S7-400 no slide acima.

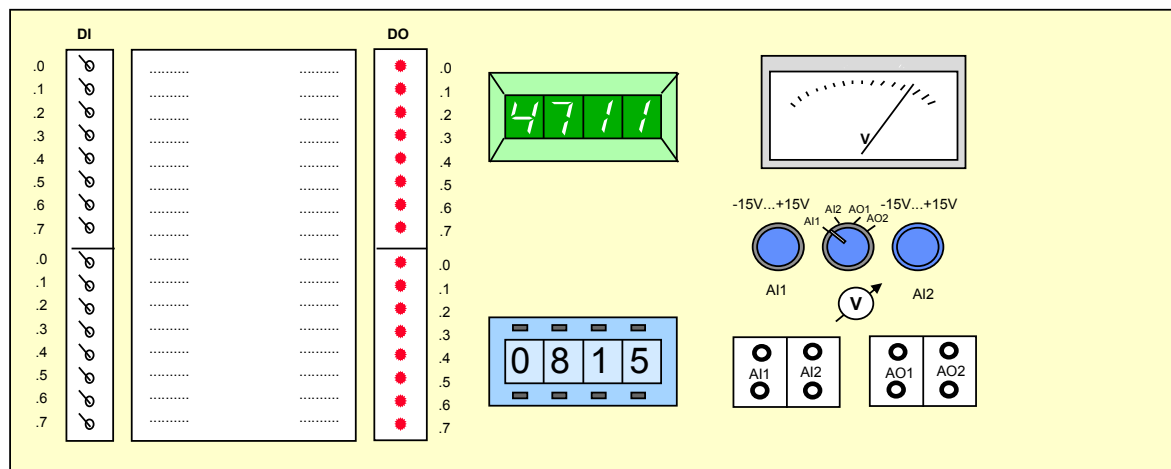
O bastidor de montagem UR 1 é configurado com os seguintes módulos:

Slot 1:	Fonte de alimentação 24V e 5V/20A	
Slot 2:	- " -	
Slot 3:	- " -	
Slot 4:	CPU 412 ou outra	
Slot 5:	vago (quando a CPU for de largura simples)	
Slot 6:	vago	
Slot 7:	vago	
Slot 8:	Entrada digital 32x24V	(do Simulador)
Slot 9:	Entrada digital 32x24V	(do modelo C. Transp.)
Slot 10:	Saída digital 32x24V 0.5A	(para Simulador)
Slot 11:	Saída digital 32x24V 0.5A	(para modelo C. Transp.)
Slot 12:	Entrada analógica 8X13 Bit	(do potenc. no Simulador)
Slot 13:	vago	
Slot 14:	vago	
Slot 15:	vago	
Slot 16:	vago	
Slot 17:	vago	
Slot 18:	vago	

### Endereçamento

Você tem os endereços padrões, conforme mostrado no slide acima, quando não foi feita nenhuma configuração ou nenhuma parametrização de novos endereços tenham sido transferidas.

## O Simulador



### SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.6Conhecimento em Automação  
Training Center

### Projeto

O Simulador é conectado às unidades de treinamento S7-300 ou S7-400 por dois cabos. Eles são divididos em três seções:

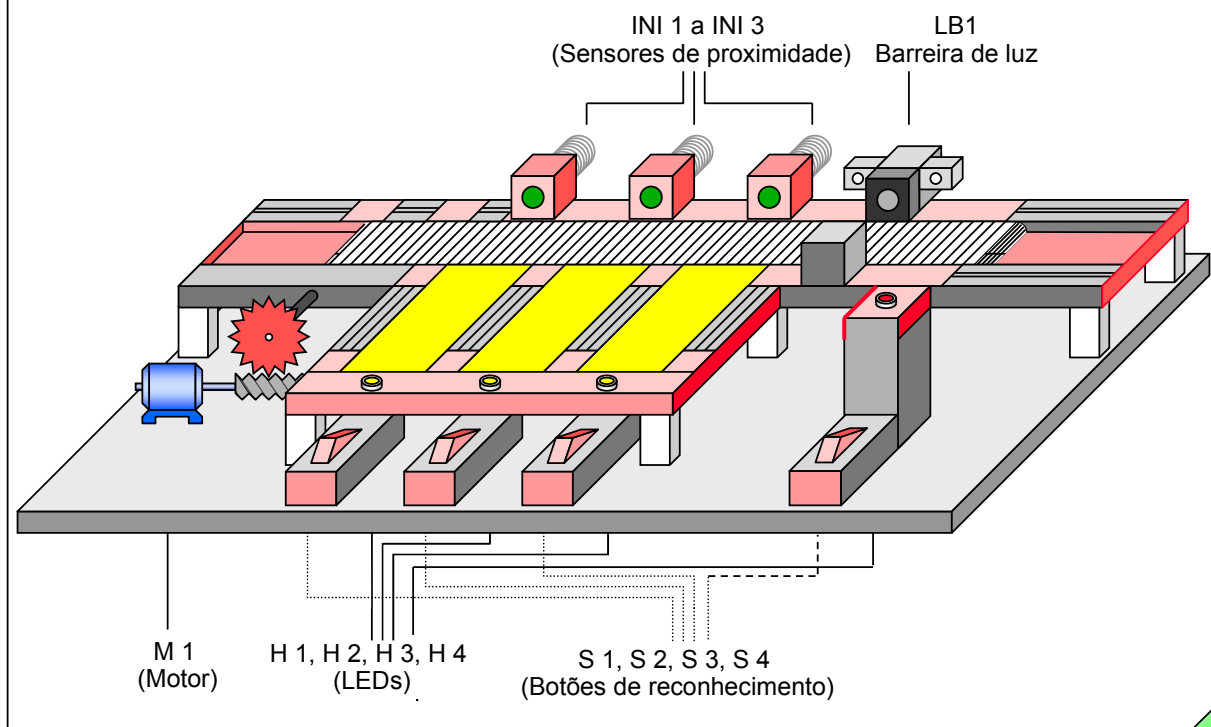
- Seção binária com 16 chaves (liga-desliga e liga com retorno por mola) e 16 LEDs.
- Seção digital com 4 chaves pushwheel (ou thumbwheel) e um display digital. Estas operam com valores BCD.
- Seção analógica com um voltímetro para mostrar os valores dos canais de saída analógicos 0 e 1. Você utiliza a chave seletora para escolher o valor de tensão que você deseja monitorar. Existem dois potenciômetros separados para ajuste de valores para os canais de entrada analógicos.

### Endereçamento

Você utiliza os seguintes endereços para endereçar as entradas e saídas em seu programa do usuário:

Sensor / Atuador	Versão A (DI16, DQ16)	Versão B (DI32, DQ32)	S7-400 (endere.padrões)
Chaves de teste	IW 0	IW 0	IW 28
LEDs	QW 8	QW 4	QW 36
Chave Pushwheel	IW 4	IW 2	IW 30
Display digital	QW 12	QW 6	QW 38
Canais analógicos	PIW 352/354	PIW 304/306	PIW 1216/1230

## O Modelo Correia Transportadora



SIMATIC S7

Siemens AG 1999. All rights reserved.

Date: 04.10.2007  
File: PRO2\_14P.7Conhecimento em Automação  
Training Center

### Projeto

O slide acima mostra um diagrama do modelo Correia Transportadora com seus sensores e atuadores.

### Addresses

S7-300 Ver. A (DI16, DO16)	S7-300 Ver. B (DI32, DO32)	S7-400 (sem Config. HW )	Sensor / Atuador	Símbolo
I 16.0	I 8.0	I 32.0	Barreira de luz LB 1	LB1
I 16.1	I 8.1	I 32.1	Ch. reconheç., posto 1	S1
I 16.2	I 8.2	I 32.2	Ch. reconheç., posto 2	S2
I 16.3	I 8.3	I 32.3	Ch. reconheç., posto 3	S3
I 16.4	I 8.4	I 32.4	Ch. reconheç., mont.final	S4
I 16.5	I 8.5	I 32.5	Sensor proximidade 1	INI1
I 16.6	I 8.6	I 32.6	Sensor proximidade 2	INI2
I 16.7	I 8.7	I 32.7	Sensor proximidade 3	INI3
Q 20.1	Q 8.1	Q 40.1	LED no posto 1	H1
Q 20.2	Q 8.2	Q 40.2	LED no posto 2	H2
Q 20.3	Q 8.3	Q 40.3	LED no posto 3	H3
Q 20.4	Q 8.4	Q 40.4	LED na montagem final	H4
Q 20.5	Q 8.5	Q 40.5	C. Transp. oper. p/ dir.	K1_CONV
Q 20.6	Q 8.6	Q 40.6	C. Transp. oper. p/ esq.	K2_CONV
Q 20.7	Q 8.7	Q 40.7	Buzina	HORN

## Solução do Exercício 1.1: Salto após uma Subtração

```
FUNCTION FC 11 : VOID
TITLE =Exercise 1.1 : salto após uma Subtração
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Chave Thumbwheel
BTD ;                // Converte formato de BCD para DINT
L    IW  0;           // Palavra de entrada 0
BTD;
-D;
JM   NEG;             // Salta, se resultado negativo
L    IW  4;
JU   END;
NEG: L    0;
END: T    QW 12;       // Display digital
END_FUNCTION
```

## Solução do Exercício 1.2: Salto após uma Multiplicação

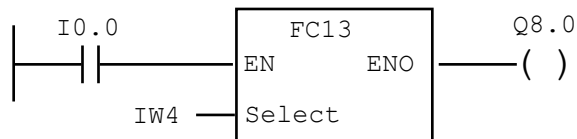
```
FUNCTION FC 12 : VOID
TITLE =Exercise 1.2 : Salto após uma Multiplicação
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Chave Thumbwheel
BTD;                  // Conversão de BCD para DINT
L    IW  0;           // Chaves de teste no Simulador
BTD;
*I;
JO   OVL;             // Salta se houver overflow
DTB;                  // Conversão de DINT para BCD
JU   END;
OVL: L    0;
END: T    QW 12;       // Display digital
END_FUNCTION
```

## Solução do Exercício 1.3: Programando um Distribuidor de Saltos

### OB1



```

FUNCTION FC 13: VOID
//Versão para SM 16Bit
// Programando um Distribuidor de Saltos
VAR_INPUT
Select: INT;
END_VAR
BEGIN
SET;
L   #Select;
AW  W#16#FF00;           // Verifica se a seleção é >255 ou
JN  Err;                 // Salta se > 255
L   #Select;             // Recarrega o valor
JL  GT5;                 // Salta p/rótulo se ACCU1-L-L >5
JU  Err;                 // Se a seleção for = 0 (não permitido)
JU  Dr_1;                // Correia p/direita (seleção=1)
JU  Dr_2;                // Correia p/esquerda (seleção=2)
JU  Dr_3;                // Parar correia (seleção=3)
JU  Ho_1;                // Liga buzina
JU  Ho_2;                // Desliga buzina
GT5:
JU  Err;
Dr_1:
S   Q   20.5;            // Correia p/direita
R   Q   20.6;
JU  End;
Dr_2:
S   Q   20.6;            // Correia p/esquerda
R   Q   20.5;
JU  End;
Dr_3:
R   Q   20.5;            // Parar Correia
R   Q   20.6;
JU  End;
Ho_1:
S   Q   20.7;            // Liga buzina
JU  End;
Ho_2:
R   Q   20.7;            // Desliga buzina
JU  End;
Err:
R   Q   20.5;            // Parar Correia
R   Q   20.6;
R   Q   20.7;           // Desliga buzina
CLR;                     // Resetar ENO
SAVE;
BE;
End:
END_FUNCTION
  
```



## Solução do Exercício 2.1: Cálculo de Expoentes

```
FUNCTION FC 21 : VOID
TITLE =Exercise 2.1: Cálculo de Expoentes
//Version for 16Bit SM
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =
L   IB   5;           // Carrega byte a direita da chave pushweel
BTI;                  // BCD para INT  -> entrada de valor
PUSH;                 // Copia ACCU1 para ACCU2
*D;                   // Forma o quadrado de valor no ACCU1
PUSH;                 // Copia quadrado do ACCU1 para ACCU2
PUSH;                 // Necessário p/S7-400: quadrado -> ACCU3
*D;                   // Forma potência de 4 no ACCU1
*D;                   // Forma potência de 6 no ACCU1
DTB;                  // Converte para BCD
T   QW  12;           // Transfere palavra baixa p/ display digital
END_FUNCTION
```

## Solução do Exercício 2.2: Troca de Dados no ACCU1

```
FUNCTION FC 22 : VOID
TITLE = Exercício 2.2: Troca de Dados no ACCU1
//Versão para SM 16Bit
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE =

L    IW  4;           // Carrega número BCD
CAW;                  // Troca dois bytes no ACCU1-L
T    QW 12;          // Mostra o resultado
END_FUNCTION
```

## Solução do Exercício 2.3: Formação de Complementos

```
FUNCTION FC 23 : VOID
TITLE = 2.3: Formação de Complementos
//Versão para SM 16Bit
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

BEGIN
NETWORK
TITLE = Complemento de Um em STL

L    IW  0;           // Carrega palavra de entrada das chaves de testes
INVI;                // Forma o complemento de um
T    QW  8;           // Transfere o resultado para os LEDs do simulador
END_FUNCTION
```

## Solução do Exercício 3.1: Calculando a Distância

```
FUNCTION FC 31 : REAL
TITLE =Exercise 3.1: Calculando a distância
AUTHOR : PT41
FAMILY : A4_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
  X1: REAL;
  Y1: REAL;
  X2: REAL;
  Y2: REAL ;
END_VAR
VAR_TEMP
  XSquare : REAL;
END_VAR
BEGIN
  NETWORK
  TITLE =
    L  #X1;           // Carrega cooredenada X de P1
    L  #X2;           // Carrega cooredenada X de P2
    -R;               // Calcula (X1-X2)
    SQR;              // Eleva (X1-X2) ao quadrado
    T  #XSquare;      // Armazena o resultado na variável TEMP
    L  #Y1;           // Carrega cooredenada Y de P1
    L  #Y2;           // Carrega cooredenada Y de P2
    -R;               // Calcula (Y1-Y2)
    SQR;              // Eleva (Y1-Y2) ao quadrado
    L  #XSquare;      // Recarrega (X1-X2) elevado ao quadrado
    +R;               // Soma
    SQRT;             // Calcula raiz quadrada
    T  #RET_VAL;      // Transfere para RET_VAL
END_FUNCTION
```

## Solução do Exercício 4.1: Programação de Loop com Endereçamento Indireto de Memória (parte 1)

```
FUNCTION FC 41 : VOID
TITLE = Exercício 4.1: Programação de Loop com Endereçamento Indireto de Memória
VAR_INPUT
    DB_Num : WORD ;
END_VAR
VAR_TEMP
    L_Counter : INT ;
    Ini_Value : REAL ;
    I_DB_Num : WORD;
    Par_Pointer : DWORD ; END_VAR
BEGIN
NETWORK
TITLE = Abre o DB

    L    #DB_Num;           // Carrega o número do DB
    T    #I_DB_Num;        // e transfere p/ variável temporária
    OPN  DB [#I_DB_Num];   // Abre DB

NETWORK
TITLE = LOOP

    L    P#0.0;             // Carrega end. do primeiro componente Tanque
    T    #Par_Pointer;      // e transfere para #T_Pointer
    L    1.0;               // Carrega constante 1,0 e
    T    #Ini_Value;        // transfere para #Ini_Value
    L    100;               // Inicializa loop de contagem com 100
    BEGN: T    #L_Counter;  // e transfere para #L_Counter
    L    #Ini_Value;
    T    DBD [#Par_Pointer]; // Transfere #Ini_Value Meas_Value[i]
    L    1.0;               // Incrementa ACCU1 (#Ini_Value)
    +R    ;                 // de 1,0
    T    #Ini_Value;        // e transfere para #Ini_Value
    L    #Par_Pointer;      // Carrega #Par_Pointer no ACCU1
    L    P#4.0;             // Incrementa o byte de endereço
    +D    ;                 // do #Par_Pointer em 4 unidades
    T    #Par_Pointer;      // e transfere resultado para #Par_Pointer
    L    #L_Counter;        // Carrega contador de loop,
    LOOP BEGN;              // Decrementa contador loop e se necessário saltar

END_FUNCTION
```

## Solução do Exercício 4.1: Programação de Loop com Endereçamento Indireto de Memória (parte 2)

```
ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;    //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)
  OB1_SCAN_1 : BYTE ;      //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)
  OB1_PRIORITY : BYTE ;    //1 (Prioridade do 1 é baixa)
  OB1_OB_NUMBR : BYTE ;    //1 (Bloco de Organização 1, OB1)
  OB1_RESERVED_1 : BYTE ;  //Reservado para o sistema
  OB1_RESERVED_2 : BYTE ;  // Reservado para o sistema
  OB1_PREV_CYCLE : INT ;   //Ciclo de tempo da varredura anterior do OB1(milisegundos)
  OB1_MIN_CYCLE : INT ;    //Mínimo cycle de tempo do OB1 (milisegundos)
  OB1_MAX_CYCLE : INT ;    //Máximo cycle de tempo do OB1 (milisegundos)
  OB1_DATE_TIME : DATE_AND_TIME ; //Data e horário da partida do OB1
END_VAR
BEGIN
NETWORK
TITLE =
  CALL FC 41 (
    DB_Num := W#16#29);
NOP 0;

END_ORGANIZATION_BLOCK
```

## Solução do Exercício 4.2: Programação de Loop com Endereçamento Indireto de Registro

FUNCTION FC 42: VOID  
TITLE = Exercício 4.2: Programação de Loop com Endereçamento Indireto de Registro  
// Versão para S7-300 e S7-400

VAR\_INPUT  
  DB\_Num : WORD ;  
END\_VAR  
VAR\_TEMP  
  I\_DB\_Num : WORD ;  
END\_VAR  
BEGIN  
NETWORK  
TITLE = Abertura do DB

  L   #DB\_Num;               // Carrega número do DB  
  T   #I\_DB\_Num;            // e transfere para variável temporária  
  OPN DB [#I\_DB\_Num];      // Abre DB

NETWORK  
TITLE = LOOP

  LAR1 P#DBX0.0;            // Carrega endereço do primeiro componente do Tanque  
  L   L#1;                  // Carrega 1 no ACCU1 (Ini\_Value.)  
  L   100;                  // 100 no ACCU1 (L\_Counter); 1 no ACCU2 (Ini\_Value)  
BEGN: TAK ;                 // L\_Counter no ACCU2, Ini\_Value no ACCU1  
  T   D [AR1,P#0.0];        // Transfere Ini\_Value para Tank[i]  
  +   L#1;                  // Incrementa Ini\_Value  
  +AR1 P#4.0;                // Incrementa AR1 de 2 unidades  
  TAK ;                     // L\_Counter no ACCU1, Ini\_Value no ACCU2  
  LOOP BEGN;                // Decrementa, verifica término e salta

END\_FUNCTION

## Solução do Exercício 4.3: Cálculo de Soma e Valor Médio

```

FUNCTION FC 43 : VOID
TITLE = Exercício 4.3: Cálculo de Soma e Valor Médio
VERSION : 0.0
VAR_INPUT
    Measured_values : ANY ;
END_VAR
VAR_OUTPUT
    Sum : REAL ;
    Mean_value : REAL ;
END_VAR
VAR_TEMP
    Num_Elements : WORD ;
    L_Counter : WORD ;
    DB_No : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =
    L   P##Measured_values; // Carrega endereço do ponteiro "ANY"
    LAR1 ; // Transfere endereço no AR1
    L   B [AR1,P#1.0]; // Carrega identificador de tipo de dado
    L   8; // Carrega identificador de REAL (16#08)
    ==I ;
    JC REAL; // Salta se tipo de dado é igual a REAL
    NOP 0; // Instruções p/ tipo de dado diferente de REAL
    CLR ; // RLO=0
    SAVE ; // BR=0
    L   L#-1; // Carrega número REAL inválido
    T   #Sum;
    T   #Mean_value;
    BEU ;
REAL: NOP 0; // Instruções para tipo de dado: REAL
    L   W [AR1,P#2.0]; // Carrega número de elementos array
    T   #Num_Elements; // Guarda número de elementos
    L   W [AR1,P#4.0]; // Carrega número do DB ou 0
    T   #DB_No; // Se: DB_No=0, então: OP DB[DB_No]=NOP
    OPN DB [#DB_No]; // Erro de Runtime!!, se DB não existe
    L   D [AR1,P#6.0]; // Carrega ponteiro para operando atual
    LAR1 ; // no AR1, Erro !! Se identificador de área é igual a "DI"
    L   0.000000e+000; // 0 no ACCU1 (Soma =0,0)
    L   #Num_Elements; // Num_Elements no ACCU1; Sum=0 no ACCU2
BEGN: T   #L_Counter; // Seta L_Counter
    TAK ; // Sum no ACCU1
    L   D [AR1,P#0.0]; // Elemento do Array no ACCU1, Sum no ACCU2
    +R ; // Sum em ACCU1
    +AR1 P#4.0; // Incrementa AR1 de 4 unidades
    L   #L_Counter; // L_Counter no ACCU1, Sum no ACCU2
    LOOP BEGN; // Decrementa e salta
    TAK ; // Sum no ACCU1
    T   #Sum; // Sum para #Sum
    L   #Num_Elements; // Sum no ACCU2, número de elementos no ACCU1
    DTR ; // Converte inteiro não sinalizado (16 bit) para REAL
    /R ; // Valor médio no ACCU1
    T   #Mean_value; // Transfere valor médio para #Mean_value
    SET ; // Seta bit BR
    SAVE ;
END_FUNCTION

```



## Solução do Exercício 5.2: Acessando Tipos de Dados Complexos

```
FUNCTION FC 52 : VOID
TITLE =Monitoração de Motores
//Versão para S7-300 e S7-400
VERSION : 0.1

VAR_INPUT
    Motor : "Motor";
END_VAR
VAR_OUTPUT
    Motor_OK : BOOL ;
    SetActDiff : DINT ;
    SetActDiffDisp : DWORD ;
END_VAR
VAR_TEMP
    SetActDifference : REAL ;
END_VAR
BEGIN
NETWORK
TITLE =

//Computando a porcentagem de desvio
    SET ;           //Obriga o first check, seta RLO p/ "1"
    SAVE ;          //Seta o bit BR p/ "1"
    L   #Motor.SetSpeed; //Carrega velocidade no ACCU1
    PUSH ;           //somente p/ S7-400, seta velocidade no ACCU2
    PUSH ;           //Carrega velocidade no ACCU3
    L   #Motor.ActualSpeed; //Seta veloc. no ACCU2, veloc. atual no ACCU1
    -R ;             //Diferença no ACCU1, seta velocidade no ACCU2
    T   #SetActDifference; //Guarda diferença na variável temporária
    TAK ;            //Diferença no ACCU2, seta velocidade no ACCU1
    /R ;             //Desvio percentual atual no ACCU1
    ABS ;            //Desvio percentual absoluto no ACCU1
    L   #Motor.SetActDiffMax; //Carrega máx. desvio percentual no ACCU1
    <=R ;             //Desvio atual igual ou menor que desvio desejado?
    AN   #Motor.Disturbance; //e nenhum distúrbio
    =    #Motor_OK;      //então Motor está OK
NETWORK
TITLE = Mostrando a diferença entre a velocidade e a velocidade atual

    L   #SetActDifference; //Carrega SetActDifference
    RND ;                  //Converte p/ DINT
    PUSH ;                 //Guarda SetActDifference no ACCU2
    DTB ;                  //Número DINT no ACCU2, número BCD no ACCU1
    JO   ERR;              //Salta se houve erro de conversão
    T   #SetActDiffDisp;   //Transfere número BCD válido p/ display digital
    TAK ;
    T   #SetActDiff;       //Transfere número DINT válido p/ #SetActDiff
    BEU ;                  //Se nenhum erro, termina
ERR: CLR ;
    SAVE ;                 //Limpa bit BR
END_FUNCTION
```

## Solução do Exercício 5.3: Leitura do Relógio do Sistema

```
FUNCTION FC 53 : VOID
TITLE = Exercício 5.3: Leitura do Relógio do Sistema
//Versão para SM 16Bit
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
  Date_Time : DATE_AND_TIME ;           //Data e hora atuais
  RET_VAL_SFC1 : INT ;                  //Retorna valor do SFC 1
END_VAR
BEGIN
NETWORK
TITLE =Call SFC 1 (READ_CLK)

  CALL SFC1 (
    RET_VAL      := #RET_VAL_SFC1,
    CDT          := #Date_Time);

  NOP 0;
NETWORK
TITLE = Mostra horas e minutos

  LAR1 P##Date_Time;                    // Captura endereço do #Date_Time
  L   LB [AR1, P#3.0];                  // Lê a hora
  T   QB 12;                            // e transfere para display digital
  L   LB [AR1, P#4.0];                  // Lê minutos
  T   QB 13;                            // e transfere para display

END_FUNCTION
```

## Solução do Exercício 6.1a: Planta de Engarrafamento – Modo de Operação

FUNCTION\_BLOCK "Mode\_Selection"

TITLE =Modo de Seleção

VERSION : 0.1

VAR\_INPUT

Start : BOOL ;

Stop : BOOL ;

Auto\_Man : BOOL ;

OM\_activate : BOOL ;

END\_VAR

VAR\_OUTPUT

Plant\_on : BOOL ;

OM\_Man : BOOL ;

OM\_Auto : BOOL ;

END\_VAR

BEGIN

NETWORK

TITLE =Planta liga / desliga

```
A  #Start;           // sinal liga planta,
S  #Plant_on;        // seta saída plant_on;
AN #Stop;            // sinal desliga planta,
R  #Plant_on;        // reseta saída plant_on;
A  #Plant_on;        //
=  #Plant_on;        //
```

NETWORK

TITLE = Modo de Operação: Manual

```
A  #Plant_on;        // se a planta estiver ligada e
AN #Auto_Man;        // se o modo manual estiver selecionado e
A  #OM_activate;     // se a entrada enter_mode estiver ativa,
S  #OM_Man;          // então seta a saída manual_mode;
A( ;
ON  #Plant_on;       // se a planta estiver desligada
O  ;                // ou
A  #Auto_Man;        // se o modo automático estiver selecionado e
A  #OM_activate;     // se o enter_mode estiver ativado,
) ;
R  #OM_Man;          // reseta a saída manual_mode;
A  #OM_Man;          //
=  #OM_Man;          //
```

NETWORK

TITLE = Modo de Operação: Automático

```
A  #Plant_on;        // se a planta estiver ligada e
A  #Auto_Man;        // se o modo automático estiver selecionado e
A  #OM_activate;     // se a entrada enter_mode estiver ativada,
S  #OM_Auto;         // então seta a saída automatic_mode;
A( ;
ON  #Plant_on;       // se a planta estiver desligada
O  ;                // ou
AN #Auto_Man;        // se o modo manual estiver selecionado e
A  #OM_activate;     // se o enter_mode estiver ativado,
) ;
R  #OM_Auto;         // reseta a saída automatic_mode;
A  #OM_Auto;         //
=  #OM_Auto;         //
```

END\_FUNCTION\_BLOCK

## Solução do Exercício 6.1b: Planta de Engarrafamento – Controle do Transportador (parte 1)

```
FUNCTION_BLOCK "Conveyor_Control"
TITLE =
VERSION : 0.1

VAR_INPUT
  OM_Man : BOOL ;
  OM_Auto : BOOL ;
  Jog_for : BOOL ;
  Jog_back : BOOL ;
  Sensor_fill : BOOL ;
  Sensor_full : BOOL ;
END_VAR
VAR_OUTPUT
  Conv_for : BOOL ;
  Conv_back : BOOL ;
  Filling_active : BOOL ;
  Full_bottles : WORD ;
END_VAR
VAR
  Filling_time : "TP";
  Bottle_counter : "CTU";
END_VAR
VAR_TEMP
  bottles : INT ;
END_VAR

BEGIN
NETWORK
TITLE = Ramificação entre o modo Manual e Automático
  SET ; // obriga o first check,
  SAVE ; // e seta o bit BR p/ "1";
  A #OM_Man; // se manual_mode estiver ativo,
  JC Man; // salta para modo manual;
  A #OM_Auto; // se automatic_mode estiver ativo,
  JC Auto; // salta para modo automático;
  R #Conv_for; // se nenhum modo de operação estiver ativo,
  R #Conv_back; // reseta o acionamento da C. Transp.,
  R #Filling_active; // reseta filling_active
  CALL #Bottle_counter (
    R := TRUE); // reseta contador

  L 0; // reseta valor de full_bottles
  T #Full_bottles;
  BEU ;

NETWORK
TITLE = Modo de Operação: Manual
//Controla a Correia Transportadora através dos botões JOG
Man: A #Jog_for;
  AN #Jog_back;
  = #Conv_for;
  A #Jog_back;
  AN #Jog_for;
  = #Conv_back;
  BEU ;
```

// (Continua na próxima página)

## Solução do Exercício 6.1b: Planta de Engarrafamento – Controle do Transportador (parte 2)

```
NETWORK
TITLE = Modo de Operação: Automático
//Partir Filling_time
Auto: A   #Sensor_fill;
      =   L   2.0;
      LD 103;
      CALL #Filling_time (
        IN      := L   2.0,
        PT      := T#3S,
        Q       := #Filling_active);

      NOP 0;

NETWORK
TITLE = Modo de Operação: Automático
//Contagem de garrafas cheias
A   #Sensor_full;
=   L   2.0;
BLD 103;
CALL #Bottle_counter (
  CU      := L   2.0,
  R       := FALSE,
  CV      := #bottles);
NOP 0;

NETWORK
TITLE = Modo de Operação: Automático
//Convertendo #bottles para BCD
//
L   #bottles;
ITB ;
T   #Full_bottles;
NOP 0;

NETWORK
TITLE = Modo de Operação: Automático
//C. Transp. p/frente enquanto o enchimento não estiver em processo
AN  #Filling_active;
=   #Conv_for;
END_FUNCTION_BLOCK
```

## Solução do Exercício 6.1b: Planta de Engarrafamento – Controle do Transportador (parte 3)

ORGANIZATION\_BLOCK "Cycle"

TITLE =

VERSION : 0.1

VAR\_TEMP

OB1\_EV\_CLASS : BYTE ; //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)

OB1\_SCAN\_1 : BYTE ; //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)

OB1\_PRIORITY : BYTE ; //1 (Prioridade do 1 é baixa)

OB1\_OB\_NUMBR : BYTE ; //1 (Bloco de Organização 1, OB1)

OB1\_RESERVED\_1 : BYTE ; //Reservado para o sistema

OB1\_RESERVED\_2 : BYTE ; // Reservado para o sistema

OB1\_PREV\_CYCLE : INT ; //Ciclo de tempo da varredura anterior do OB1(milisegundos)

OB1\_MIN\_CYCLE : INT ; //Mínimo cycle de tempo do OB1 (milisegundos)

OB1\_MAX\_CYCLE : INT ; //Máximo cycle de tempo do OB1 (milisegundos)

OB1\_DATE\_TIME : DATE\_AND\_TIME ; //Data e horário da partida do OB1

Full\_bottles : INT ;

END\_VAR

BEGIN

NETWORK

TITLE = Modo de Operação

//Controle do Modo de Operação

A "Start";

= L 22.0;

BLD 103;

A "Stop";

= L 22.1;

BLD 103;

A "Man/Auto";

= L 22.2;

BLD 103;

A "Enter\_Mode";

= L 22.3;

BLD 103;

CALL "Mode\_selection" , "Mode\_Selection\_DB" (

Start := L 22.0,

Stop := L 22.1,

Auto\_Man := L 22.2,

OM\_activate := L 22.3,

Plant\_on := "Plant\_on",

OM\_Man := "Manual\_Mode",

OM\_Auto := "Automatic\_Mode");

NOP 0;

// (Continua na próxima página)

## Solução do Exercício 6.1b: Planta de Engarrafamento – Controle do Transportador (parte 4)

### NETWORK

TITLE = Controlando a Correia Transportadora

```
A  "Manual_Mode";
=  L  22.0;
BLD 103;
A  "Automatic_Mode";
=  L  22.1;
BLD 103;
A  "Jog_forward";
=  L  22.2;
BLD 103;
A  "Jog_backward";
=  L  22.3;
BLD 103;
A  "Filling_Position";
=  L  22.4;
BLD 103;
A  "Counting_Bottles";
=  L  22.5;
BLD 103;
CALL "Conveyor_Control" , "Conveyor_Control_DB" (
    OM_Man      := L  22.0,
    OM_Auto     := L  22.1,
    Jog_for     := L  22.2,
    Jog_back    := L  22.3,
    Sensor_fill := L  22.4,
    Sensor_full := L  22.5,
    Conv_for    := "Conveyor_forward",
    Conv_back   := "Conveyor_backward",
    Filling_active := "Filling_in_progress",
    Full_bottles := "Display");
NOP 0;
```

END\_ORGANIZATION\_BLOCK

## Solução do Exercício 6.2a: FB1 para Estação de Trabalho (parte 1)

```
FUNCTION_BLOCK "Station"
TITLE = Controlando uma estação de trabalho
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
    Initial : BOOL ;
    Proxy_switch : BOOL ;
    Acknowledge : BOOL ;
    Clock_bit_q : BOOL ;
    Clock_bit_s : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Transp_req : BOOL ;
END_VAR
VAR_IN_OUT
    Conv_busy : BOOL ;
END_VAR
VAR
    State : STRUCT
        Process_piece : BOOL ;
        Piece_finished : BOOL ;
        Place_piece_on_conv : BOOL ;
        Wait_for_piece : BOOL ;
        Take_piece_from_conv : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
TITLE = Inicialização
//Por meio da entrada "Initial" o estado básico #Process_piece é setado
    A   #Initial;
    S   #State.Process_piece;
    R   #State.Piece_finished;
    R   #State.Place_piece_on_conv;
    R   #State.Wait_for_piece;
    R   #State.Take_piece_from_conv;
    R   #Conv_busy;

NETWORK
TITLE = Estado: Process_piece
//Neste estado a peça bruta é processada. Processamento é terminado
// quando o operador reconhece o término da peça bruta
//por meio do botão "S1"
    AN  #State.Process_piece;
    JC  Pfin;
    S   #LED;                //o LED fica ligado constantemente;
    R   #Transp_req;
    A   #Acknowledge;        //quando o operador reconhece,
    R   #State.Process_piece; //uma mudança de estado ocorre;
    R   #LED;
    S   #State.Piece_finished;

// (Continua na próxima página)
```



## Solução do Exercício 6.2a: FB1 para Estação de Trabalho (parte 2)

NETWORK

TITLE = Estado: Piece\_finished

//No estado #Piece\_finished o operador espera pela permissão

//para colocar a peça bruta na C. Transp.. O sinal #Conv\_busy indica,

//se a C.Transp. está ocupada ou não. Quando a CT estiver livre, uma mudança de estado

//para o estado Place\_piece\_on\_conv é realizada.

Pfin: AN #State.Piece\_finished;

JC PpCo;

A #Clock\_bit\_s; //pisca lento;

= #LED;

AN #Conv\_busy; //quando a C. Transp. estiver livre,

S #Conv\_busy; //isto é sinalizado ocupado

R #LED; //então uma mudança de estado é realizada;

R #State.Piece\_finished;

S #State.Place\_piece\_on\_conv;

NETWORK

TITLE = Estado: Place\_piece\_on\_conv

PpCo: AN #State.Place\_piece\_on\_conv;

JC Wait;

A #Clock\_bit\_q; //pisca lento;

= #LED;

A #Proxy\_switch; //quando a peça estiver colocada na Correia,

S #Transp\_req; //o transportador parte,

R #LED; //e o LED é apagado;

A #Transp\_req; //Quando a Correia estiver movimentando,

AN #Proxy\_switch; //então a peça bruta deixa a chave de proximidade,

R #State.Place\_piece\_on\_conv; // e uma mudança de estado é realizada;

S #State.Wait\_for\_piece;

NETWORK

TITLE = Estado: Wait\_for\_piece

//Esperando por uma nova peça bruta. A chegada de uma nova peça é iniciada pelo

//sensor de proximidade da Correia

Wait: AN #State.Wait\_for\_piece;

JC TpCo;

R #LED; //o LED é desligado;

A #Proxy\_switch; //uma nova peça bruta chega,

R #Transp\_req; //a C. Transp. é parada,

R #State.Wait\_for\_piece; //e uma mudança de estado é realizada;

S #State.Take\_piece\_from\_conv;

NETWORK

TITLE = Estado: Take\_piece\_from\_conv

//Neste novo estado a nova peça bruta é pega da C. Transp. paraa

//o posto de trabalho

TpCo: AN #State.Take\_piece\_from\_conv;

JC END;

A #Clock\_bit\_q; //o LED pisca rapidamente

= #LED;

AN #Proxy\_switch; //quando a nova peça bruta é pega da C. Transp.,

R #Conv\_busy; //a C. Transp. fica livre,

R #LED; //o LED é desligado

R #State.Take\_piece\_from\_conv; //e uma mudança de estado é realizada;

S #State.Process\_piece;

END: BEU ;

END\_FUNCTION\_BLOCK

## Solução do Exercício 6.2a: FB2 para o Transportador (parte 1)

```
FUNCTION_BLOCK "Transport"
TITLE = Controlando a Correia Transportadora
VERSION : 0.1
VAR_INPUT
    Initial : BOOL ;
    L_Barrier : BOOL ;
    Acknowledge : BOOL ;
    Transp_req : BOOL ;
    Clock_Bit : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Conv_right : BOOL ;
    Conv_left : BOOL ;
END_VAR
VAR
    State : STRUCT
        Waiting : BOOL ;
        Conv_right : BOOL ;
        Assembly : BOOL ;
        Conv_left : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
    TITLE = Inicialização
        A    #Initial;
        S    #State.Waiting;
        R    #State.Conv_right;
        R    #State.Assembly;
        R    #State.Conv_left;
NETWORK
    TITLE = Estado: Esperando
//A C. Transp. espera neste estado pela finalização da peça.
        AN  #State.Waiting;
        JC  RECH;
        R   #Conv_right;
        R   #Conv_left;
        R   #LED;
        A   #Transp_req;
        R   #State.Waiting;
        S   #State.Conv_right;
NETWORK
    TITLE = Estado: Conv_right
//Este estado descreve o transporte da peça acabada em direção a montagem final
    RECH: AN  #State.Conv_right;
        JC  ENDM;
        S   #Conv_right;
        A   #Clock_Bit;
        =   #LED;
        AN  #L_Barrier;
        R   #Conv_right;
        R   #State.Conv_right;
        S   #State.Assembly;
        AN  #L_Barrier;
        =   #L_Barrier;
```

// (Continua na próxima página)

## Solução do Exercício 6.2a: FB2 para o Transportador (parte 2)

NETWORK

TITLE = Estado: Montagem

//Neste estado, a peça finalizada é removida e uma nova peça bruta é deixada

//na correia. Depois disto, o transporte da peça bruta em direção da estação de

//processamento vazia é iniciado com S4.

//

ENDM: AN #State.Assembly;

JC LINK;

S #LED;

A #Acknowledge;

R #LED;

R #State.Assembly;

S #State.Conv\_left;

NETWORK

TITLE = Estado: Conv\_left

//Neste estado, o transporte da nova peça bruta para a estação ocorre, que entrega

//a peça finalizada.

LINK: AN #State.Conv\_left;

JC ENDE;

S #Conv\_left;

A #Clock\_Bit;

= #LED;

AN #Transp\_req;

R #Conv\_left;

R #State.Conv\_left;

S #State.Waiting;

ENDE: BEU ;

END\_FUNCTION\_BLOCK

## Solução do Exercício 6.2a: OB1

ORGANIZATION\_BLOCK "Cycle"

TITLE = "Varredura do Programa Principal (Ciclo)"

VERSION : 0.1

VAR\_TEMP

OB1\_EV\_CLASS : BYTE ; //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)

OB1\_SCAN\_1 : BYTE ; //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)

OB1\_PRIORITY : BYTE ; //1 (Prioridade do 1 é baixa)

OB1\_OB\_NUMBR : BYTE ; //1 (Bloco de Organização 1, OB1)

OB1\_RESERVED\_1 : BYTE ; //Reservado para o sistema

OB1\_RESERVED\_2 : BYTE ; // Reservado para o sistema

OB1\_PREV\_CYCLE : INT ; //Ciclo de tempo da varredura anterior do OB1(milisegundos)

OB1\_MIN\_CYCLE : INT ; //Mínimo cycle de tempo do OB1 (milisegundos)

OB1\_MAX\_CYCLE : INT ; //Máximo cycle de tempo do OB1 (milisegundos)

OB1\_DATE\_TIME : DATE\_AND\_TIME ; //Data e horário da partida do OB1

END\_VAR

BEGIN

NETWORK

TITLE = Convocação do bloco de controle da estação

CALL "Station" , "Station\_DB" (

Initial := "INITIALIZATION",

Proxy\_switch := "IN1",

Acknowledge := "S1",

Clock\_bit\_q := "CLOCK\_BIT\_FAST",

Clock\_bit\_s := "CLOCK\_BIT\_SLOW",

LED := "H1",

Transp\_req := "Transport\_DB".Transp\_req);

NETWORK

TITLE = Convocação do bloco de transporte da estação

CALL "Transport" , "Transport\_DB" (

Initial := "INITIALIZATION",

L\_Barrier := "LB1",

Acknowledge := "S4",

Clock\_bit := "CLOCK\_BIT\_FAST",

LED := "H4",

Conv\_right := "K1\_CONVR",

Conv\_left := "K2\_CONVL");

END\_ORGANIZATION\_BLOCK

## Solução do Exercício 6.2b: Extensão para 3 Estações (FB10, parte 1)

```
FUNCTION_BLOCK "ASSEMBLY_LINE"
TITLE =
VERSION : 0.1

VAR
  Station_1 : "STATION";
  Station_2 : "STATION";
  Station_3 : "STATION";
  Transport : "TRANSPORT";
  Conv_busy : BOOL ;
END_VAR
VAR_TEMP
  trans_1 : BOOL ;
  trans_2 : BOOL ;
  trans_3 : BOOL ;
  trans : BOOL ;
END_VAR

BEGIN
NETWORK
TITLE = Convocação Station_1(Estação 1)
  A  "INITIALIZATION";
  =  L   1.0;
  BLD 103;
  A  "INI1";
  =  L   1.1;
  BLD 103;
  A  "S1";
  =  L   1.2;
  BLD 103;
  A  "CLOCK_BIT_FAST";
  =  L   1.3;
  BLD 103;
  A  "CLOCK_BIT_SLOW";
  =  L   1.4;
  BLD 103;
  CALL #Station_1 (
    Initial      := L   1.0,
    Proxy_switch := L   1.1,
    Acknowledge  := L   1.2,
    Clock_bit_q  := L   1.3,
    Clock_bit_s  := L   1.4,
    LED          := "H1",
    Transp_req   := #trans_1,
    Conv_busy    := #Conv_busy);
  NOP 0;
```

// (Continua na próxima página)

## Solução do Exercício 6.2b: Extensão para 3 Estações (FB10, parte 2)

```
NETWORK
TITLE = Convocação Station_2 (Estação 2)
A  "INITIALIZATION";
=  L  1.0;
BLD 103;
A  "INI2";
=  L  1.1;
BLD 103;
A  "S2";
=  L  1.2;
BLD 103;
A  "CLOCK_BIT_FAST";
=  L  1.3;
BLD 103;
A  "CLOCK_BIT_SLOW";
=  L  1.4;
BLD 103;
CALL #Station_2 (
    Initial      := L  1.0,
    Proxy_switch := L  1.1,
    Acknowledge  := L  1.2,
    Clock_bit_q  := L  1.3,
    Clock_bit_s  := L  1.4,
    LED          := "H2",
    Transp_req   := #trans_2,
    Conv_busy    := #Conv_busy);
NOP 0;
```

```
NETWORK
TITLE = Convocação Station_3 (Estação 3)
A  "INITIALIZATION";
=  L  1.0;
BLD 103;
A  "INI3";
=  L  1.1;
BLD 103;
A  "S3";
=  L  1.2;
BLD 103;
A  "CLOCK_BIT_FAST";
=  L  1.3;
BLD 103;
A  "CLOCK_BIT_SLOW";
=  L  1.4;
BLD 103;
CALL #Station_3 (
    Initial      := L  1.0,
    Proxy_switch := L  1.1,
    Acknowledge  := L  1.2,
    Clock_bit_q  := L  1.3,
    Clock_bit_s  := L  1.4,
    LED          := "H3",
    Transp_req   := #trans_3,
    Conv_busy    := #Conv_busy);
NOP 0;
```

// (Continua na próxima página)

## Solução do Exercício 6.2b: Extensão para 3 Estações (FB10, parte 3)

NETWORK  
TITLE = Lincando as saídas às entradas

```
O  #trans_1;  
O  #trans_2;  
O  #trans_3;  
=  #trans;
```

NETWORK  
TITLE = Convocação Transport (Transporte)

```
A  "INITIALIZATION";  
=  L    1.0;  
BLD 103;  
A  "LB1";  
=  L    1.1;  
BLD 103;  
A  "S4";  
=  L    1.2;  
BLD 103;  
A  #trans;  
=  L    1.3;  
BLD 103;  
A  "CLOCK_BIT_FAST";  
=  L    1.4;  
BLD 103;  
CALL #Transport (  
    Initial      := L    1.0,  
    L_Barrier    := L    1.1,  
    Acknowledge  := L    1.2,  
    Transp_req   := L    1.3,  
    Clock_Bit    := L    1.4,  
    LED          := "H4",  
    Conv_right   := "K1_CONVR",  
    Conv_left    := "K2_CONVL");  
NOP 0;  
END_FUNCTION_BLOCK
```

## Solução do Exercício 6.2b: Extensão para 3 Estações (OB1)

```
ORGANIZATION_BLOCK "CYCLE"
```

```
TITLE =
```

```
VERSION : 0.1
```

```
VAR_TEMP
```

```
OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)
```

```
OB1_SCAN_1 : BYTE ;        //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)
```

```
OB1_PRIORITY : BYTE ;      //1 (Prioridade do 1 é baixa)
```

```
OB1_OB_NUMBR : BYTE ;      //1 (Bloco de Organização 1, OB1)
```

```
OB1_RESERVED_1 : BYTE ;    //Reservado para o sistema
```

```
OB1_RESERVED_2 : BYTE ;    // Reservado para o sistema
```

```
OB1_PREV_CYCLE : INT ;     //Ciclo de tempo da varredura anterior do OB1(milisegundos)
```

```
OB1_MIN_CYCLE : INT ;      //Mínimo cicle de tempo do OB1 (milisegundos)
```

```
OB1_MAX_CYCLE : INT ;      //Máximo cicle de tempo do OB1 (milisegundos)
```

```
OB1_DATE_TIME : DATE_AND_TIME ; //Data e horário da partida do OB1
```

```
END_VAR
```

```
BEGIN
```

```
NETWORK
```

```
TITLE = Convocação Assembly Line (Linha de Montagem)
```

```
    CALL "ASSEMBLY_LINE" , "ASSEMBLY_LINE_DB" ;
```

```
    NOP 0;
```

```
END_ORGANIZATION_BLOCK
```



## Solução do Exercício 7.2: Testando o Bloco de Dados (SFC 24: somente S7 400)

```

FUNCTION FC 72 : INT
TITLE = Exercício 7.2: Testando o Bloco de Dados (somente S7-400)
VERSION : 0.1

VAR_INPUT
    DB_NUM : WORD ;
END_VAR
VAR_TEMP
    I_DB_Length : WORD ;
    I_RET_VAL : INT ;
    I_Write_Protect : BOOL ;
END_VAR
BEGIN
NETWORK
TITLE = Testando DB
//somente para S7-400

    CALL SFC 24 (
        DB_NUMBER      := #DB_NUM,
        RET_VAL         := #I_RET_VAL,
        DB_LENGTH       := #I_DB_Length,
        WRITE_PROT      := #I_Write_Protect);
    L   #I_RET_VAL;
    L   W#16#0;
    ==I ;
    JC  DBOK;                // DB disponível na memória de trabalho
    TAK ;
    L   W#16#80A1;
    ==I ;
    JC  NODB;                // DB não disponível na CPU
    TAK ;
    L   W#16#80B1;
    ==I ;
    JC  NODB;                // DB disponível na memória de trabalho
    TAK ;
    L   W#16#80B2;
    ==I ;
    JC  DBLM;                // DB somente na memória de carga
    NODB: L   -1;
    T   #RET_VAL;           // DB não disponível na CPU
    BEU ;
    DBLM: L   1;
    T   #RET_VAL;           // DB somente na memória de carga
    BEU ;
    DBOK: L   0;
    T   #RET_VAL;           // DB disponível memória de trabalho

END_FUNCTION

```

## Solução do Exercício 7.3: Criando um DB (SFC 22)

ORGANIZATION\_BLOCK OB 100

TITLE = Exercício 7.3: Criando um DB

//Versão para S7-400

VERSION : 0.1

VAR\_TEMP

OB100\_EV\_CLASS : BYTE ; //16#13, Evento Classe 1, entrando estado evento, evento

// logado no buffer de diagnóstico

OB100\_STRTUP : BYTE ; //16#81/82/83/84 Método de partida (startup)

OB100\_PRIORITY : BYTE ; //27 (Prioridade de 1 é mais baixa)

OB100\_OB\_NUMBR : BYTE ; //100 (Bloco de Organização 100, OB100)

OB100\_RESERVED\_1 : BYTE ; //Reservado para sistema

OB100\_RESERVED\_2 : BYTE ; //Reservado para sistema

OB100\_STOP : WORD ; //Evento que causou stop da CPU (16#4xxx)

OB100\_STRT\_INFO : DWORD ; //Informação de como partiu o sistema

OB100\_DATE\_TIME : DATE\_AND\_TIME ; //Data e hora que o OB100 partiu

END\_VAR

BEGIN

NETWORK

TITLE = Criando o DB10

CALL SFC 22(

LOW\_LIMIT := W#16#A, // identificação com decimal 10 (DB10)

UP\_LIMIT := W#16#A, // "

COUNT := W#16#28, // identificação com decimal 40 ( 40 Bytes)

RET\_VAL := MW 0,

DB\_NUMBER := QW 38);

END\_ORGANIZATION\_BLOCK

## Solução do Exercício 7.4: Copiando DB da Memória de Carga para de Trabalho (SFC 20)

ORGANIZATION\_BLOCK OB 1

TITLE = Exercício 7.4: Copiando DB da Memória de Carga para de Trabalho

//Versão para S7-400

VERSION : 2.10

VAR\_TEMP

OB1\_EV\_CLASS : BYTE ; //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)

OB1\_SCAN\_1 : BYTE ; //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)

OB1\_PRIORITY : BYTE ; //1 (Prioridade do 1 é baixa)

OB1\_OB\_NUMBR : BYTE ; //1 (Bloco de Organização 1, OB1)

OB1\_RESERVED\_1 : BYTE ; //Reservado para o sistema

OB1\_RESERVED\_2 : BYTE ; // Reservado para o sistema

OB1\_PREV\_CYCLE : INT ; //Ciclo de tempo da varredura anterior do OB1(milisegundos)

OB1\_MIN\_CYCLE : INT ; //Mínimo cicle de tempo do OB1 (milisegundos)

OB1\_MAX\_CYCLE : INT ; //Máximo cicle de tempo do OB1 (milisegundos)

OB1\_DATE\_TIME : DATE\_AND\_TIME ; //Data e horário da partida do OB1

END\_VAR

BEGIN

NETWORK

TITLE =

A I 28.0;

FP M 0.0;

JNB \_001;

CALL SFC 20 (

SRCLBL := P#DB20.DBX 0.0 BYTE 40,

RET\_VAL := QW 38,

DSTBLK := P#DB10.DBX 0.0 BYTE 40);

\_001: NOP 0;

END\_ORGANIZATION\_BLOCK

## Solução do Exercício 7.5: Inicializando um DB com "0" (SFC 21: FILL)

```

FUNCTION FC 75 : BOOL
TITLE = Exercício 7.5: Inicializando um DB com "0" (somente S7-400)
VERSION : 0.1
VAR_INPUT
    DB_NUM : WORD ;
    INI : BYTE ;
END_VAR
VAR_TEMP
    I_RET_VAL : INT ;
    I_DB_Length : WORD ;
    I_WRITE_PROT : BOOL ;
    I_ANY : ANY ;
    DB_No : WORD ;
    I_INI : BYTE ;
    I_RET_VAL1 : INT ;
END_VAR
BEGIN
NETWORK
TITLE =
//Verifica se o DB está na memória de trabalho
CALL "TEST_DB" (
    DB_NUMBER      := #DB_NUM,
    RET_VAL        := #I_RET_VAL,
    DB_LENGTH      := #I_DB_Length,
    WRITE_PROT     := #I_WRITE_PROT);

L   #I_RET_VAL;
L   W#16#0;
==I ;                               // DB na memória de trabalho
AN  #I_WRITE_PROT;
JC  OK;
CLR ;                               // Não é possível inicialização
=   #RET_VAL;                       // Retorna FALSE
BEU ;
OK: LAR1 P##I_ANY;                  // Atribui variável temp. ANY
L   B#16#10;                        // Identificador para ANY
T   LB [AR1,P#0.0];                 // para Byte-Offset 0
L   B#16#2;                         // Identificador para tipo de dado BYTE
T   LB [AR1,P#1.0];                 // para Byte-Offset 1
L   #I_DB_Length;                  // carrega comprimento do DB em bytes
T   LW [AR1,P#2.0];                 // para Byte-Offset 2
L   #DB_NUM;                       // carrega número do DB
T   LW [AR1,P#4.0];                 // para Byte-Offset 4
L   P#DBX 0.0;                     // carrega ponteiro para DBX0.0
T   LD [AR1,P#6.0];                 // para Byte-Offset 6
L   #INI;                           // Byte de inicialização
T   #I_INI;                         // na variável temp.

CALL SFC 21 (
    BVAL      := #I_INI, // somente possível com variável temp.
    RET_VAL   := #I_RET_VAL,
    BLK       := #I_ANY);
SET ;
=   #RET_VAL;
BE  ;

END_FUNCTION

```

## Solução do Exercício 7.6: Escrevendo uma Mensagem no Buffer de Diagnóstico (SFC 52)

```
FUNCTION FC 76 : VOID
TITLE =
//Exercício 7.6: Escrevendo uma Mensagem no Buffer de Diagnóstico (SFC 52)
//Versão para S7-300 SM 16 Bit
VERSION : 0.0
VAR_TEMP
  I_RET_VAL : INT ;
  info1 : WORD ;
  info2 : DWORD ;
END_VAR

BEGIN
NETWORK
TITLE =

  L   W#16#8;
  T   #info1;
  L   W#16#1;
  T   #info2;

  CALL "WR_USMSG" (
    SEND      := TRUE,
    EVENTN    := W#16#9B0A,
    INFO1     := #info1,
    INFO2     := #info2,
    RET_VAL   := #I_RET_VAL);

END_FUNCTION
```

## Solução do Exercício 7.7: Bloco Contador com "Debouncing de Contato"

```
FUNCTION_BLOCK FB 71
TITLE = Exercício 7.7: Bloco Contador com "Debouncing de Contato"
//Versão para S7-300 SM 16 bit
VERSION : 0.1

VAR_INPUT
  CU : BOOL ;
  R : BOOL ;
  PV : INT ;
  PT : TIME ;
END_VAR
VAR_OUTPUT
  Q : BOOL ;
  CV : INT ;
END_VAR
VAR
  Pulse_Counter : "CTU";
  Pulse_Time : "TON";
END_VAR
VAR_TEMP
  Edge_memory : BOOL ;
END_VAR

BEGIN
NETWORK
TITLE =
  A   #Edge_memory;
  =   L   1.0;
  BLD 103;
  A   #R;
  =   L   1.1;
  BLD 103;
  A( ;
  A   #CU;
  =   L   1.2;
  BLD 103;
  CALL #Pulse_Time (
    IN      := L   1.2,
    PT      := #PT,
    Q        := #Edge_memory);

  A   BR;
  ) ;
  JNB _001;
  CALL #Pulse_Counter (
    CU      := L   1.0,
    R        := L   1.1,
    PV      := #PV,
    Q        := #Q,
    CV      := #CV);
_001: NOP 0;

END_FUNCTION_BLOCK
```

## Solução do Exercício 8.1: Manipulação de Erro no FC43 (parte 1)

```

FUNCTION FC 81 : INT
TITLE =Exercise 8.1: Calculation of sum, mean value with error handling
// Solution for S7-300/400
VERSION : 0.0
VAR_INPUT
    Measured_values : ANY ;
END_VAR
VAR_OUTPUT
    Sum : REAL ;
    Mean_value : REAL ;
END_VAR
VAR_TEMP
    Num_Elements : WORD ;
    L_Counter : WORD ;
    DB_No : WORD ;
    Sum_1 : REAL ;
    sfc_ret_val : INT ;
    sfc_prgflt : DWORD ;
    sfc_accflt : DWORD ;
    I_BR : BOOL ;           //usuário bit BR
END_VAR
BEGIN
NETWORK
TITLE =
    L   P##Measured_values; // Carrega ponteiro de área em pont. "ANY"
    LAR1 ;                  // Ponteiro de área em AR1
    L   B [AR1,P#1.0];      // Lê identificador de tipo de dado
    L   8;                  // Carrega identificador REAL (16#08)
    ==I ;
    L   -1;                 // Identificador para tipo de dado difente de REAL
    JCN ERRO;              // Salta se tipo de dado diferente deREAL

// Os seguintes eventos são mascarados:
// Número de falha de um DB Global
// Número de falha de um DB Instance,
// Erro de área na leitura,
// Erro de comprimento de área na leitura
    CALL SFC 36 (
        PRGFLT_SET_MASK := DW#16#40C0014,
        ACCFLT_SET_MASK := DW#16#0,
        RET_VAL         := #sfc_ret_val,
        PRGFLT_MASKED   := #sfc_prgflt,
        ACCFLT_MASKED   := #sfc_accflt);

    L   W [AR1,P#2.0];      // Carrega número de elementos do array
    T   #Num_Elements;      // Inicializa contador de loop
    L   W [AR1,P#4.0];      // Carrega número do DB ou 0
    T   #DB_No;            // Se: DB_No=0 então: OPN DB[DB_No]=NOP
    OPN DB [#DB_No];       // Erro Run-time está agora mascarado
    L   D [AR1,P#6.0];      // Carrega ponteiro de área para endereço atual
    LAR1 ;                 // no AR1, erro!! Se identificador de área "DI"
    L   0.000000e+000;      // 0 para Accu1 (Sum =0.0)
    L   #Num_Elements;      // Contador para ACCU1; Sum=0 para ACCU2
    BEGIN: T   #L_Counter;  // Seta L_Counter
    TAK ;                  // Sum no ACCU1
    L   D [AR1,P#0.0];      // Elemento array no ACCU1, Sum no ACCU2
    +R ;                   // Sum no ACCU1
    +AR1 P#4.0;            // Incrementa AR1 de 4 unidades

// (Continua na próxima página)

```

## Solução do Exercício 8.1: Manipulação de Erro no FC43 (parte 2)

```

L   #L_Counter;           // L_Counter no ACCU1, Sum no ACCU2
LOOP_BEGN;                // Decrementa e salta
TAK ;                     // Sum no ACCU1
T   #Sum_1;               // Sum??? para #Sum_1
// Avaliação de erro
CALL SFC 38 (
    PRGFLT_QUERY          := DW#16#40C0014,
    ACCFLT_QUERY          := DW#16#0,
    RET_VAL               := #sfc_ret_val,
    PRGFLT_CLR            := #sfc_prgflt,
    ACCFLT_CLR            := #sfc_accflt);

L   #sfc_prgflt;          // Verifica falha no DB
L   DW#16#40C0000;
AD  ;                     // Bitwise "Roundup"
L   -2;                   // Código de erro para DB não existe
JN  ERRO;                 // salta se erro
L   #sfc_prgflt;          // Verifica para erro de área ou comprimento de área
L   DW#16#14;
AD  ;
L   -4;                   // Identificador para erro de área ou comprimento de área
JN  ERRO;                 // Salta se erro
//
// nenhum erro ocorrido, procede com processamento "normal"
L   #Sum_1;
T   #Sum;                 // Atribui parâmetro #Sum
L   #Num_Elements;        // Sum no ACCU2, número no ACCU1
DTR ;                     // Inteiro não sinalizado (16 Bit) para REAL
/R  ;                     // Valor médio no ACCU1
T   #Mean_value;          // Valor médio para #Mean_value
SET ;                     // Seta bit BR em 1
=   #I_BR ;
L   0;                    // Todos os identificadores O.K.
T   RET_VAL;
JU  DMSK;                 // Salta para desmascaramento de erro síncrono
//
// Avaliação de erro
//
ERRO: CLR ;               // Instruções no caso do erro RLO=0
=   #I_BR ;               // BR =0
T   #RET_VAL;             // Transfere código de erro para RET_VAL
L   L#-1;                 // Carrega número Real inválido
T   #Sum;
T   #Mean_value;
DMSK: NOP 0;              // Demascara falha síncrona
CALL SFC 37 (
    PRGFLT_RESET_MASK     := DW#16#40C0014,
    ACCFLT_RESET_MASK     := DW#16#0,
    RET_VAL               := #sfc_ret_val,
    PRGFLT_MASKED         := #sfc_prgflt,
    ACCFLT_MASKED         := #sfc_accflt);
CLR ;                     // Obriga o first check, RLO = 0
A   #I_BR ;               // Copia bit BR do usuário
SAVE ;                    // Coloca bit BR no sistema
BEU ;
END_FUNCTION

```



## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB1, parte 1)

```
FUNCTION_BLOCK "Station"
TITLE = Controlando uma estação de trabalho
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_INPUT
    Initial : BOOL ;
    Proxy_switch : BOOL ;
    Acknowledge : BOOL ;
    Clock_bit_q : BOOL ;
    Clock_bit_s : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Transp_req : BOOL ;
END_VAR
VAR_IN_OUT
    Conv_busy : BOOL ;
END_VAR
VAR
    State : STRUCT
        Process_piece : BOOL ;
        Piece_finished : BOOL ;
        Place_piece_on_conv : BOOL ;
        Wait_for_piece : BOOL ;
        Take_piece_from_conv : BOOL ;
    END_STRUCT ;
END_VAR
BEGIN
NETWORK
TITLE = Inicialização
//Por meio da entrada "Initial" o estado básico #Process_piece é setado
    A   #Initial;
    S   #State.Process_piece;
    R   #State.Piece_finished;
    R   #State.Place_piece_on_conv;
    R   #State.Wait_for_piece;
    R   #State.Take_piece_from_conv;
    R   #Conv_busy;

NETWORK
TITLE = Estado: Process_piece
//Neste estado a peça bruta é processada. O processamento é terminado
//quando o operador reconhece o término da peça bruta
//por meio do botão "S1"
    AN  #State.Process_piece;
    JC  Pfin;
    S   #LED;                //O fica ligado LED permanentemente ;
    R   #Transp_req;
    A   #Acknowledge;        //quando o operador reconheces,
    R   #State.Process_piece; //uma mudança de estado ocorre;
    R   #LED;
    S   #State.Piece_finished;

// (Continua na próxima página)
```

## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB1, parte 2)

### NETWORK

TITLE = Estado: Piece\_finished

//Neste estado #Piece\_finished o operador espera pela permissão  
//para colocar a peça bruta na C. Transp.. O sinal #Conv\_busy indica,  
//se a C. Transp. está ocupada ou não. Quando a C. Transp. está livre, mudança de estado  
//para o estado Place\_piece\_on\_conv é realizada.

```
Pfin: AN  #State.Piece_finished;
      JC  PpCo;
      A   #Clock_bit_s;          //pisca lento;
      =   #LED;
      AN  #Conv_busy;           //quando a C. Transp. está livre,
      S   #Conv_busy;           //este é marcado ocupado
      R   #LED;                 //uma mudança de estado é realizada;
      R   #State.Piece_finished;
      S   #State.Place_piece_on_conv;
```

### NETWORK

TITLE = Estado: Place\_piece\_on\_conv

PpCo: AN #State.Place\_piece\_on\_conv;

```
JC  Wait;
A   #Clock_bit_q;          //pisca rápido;
=   #LED;
A   #Proxy_switch;         //Quando a peça é colocada na correia,
S   #Transp_req;           //o transportador parte,
R   #LED;                  //e o LED é apagado;
A   #Transp_req;           //Quando a correia é movida,
AN  #Proxy_switch;         //a peça bruta sai da frente do sensor de proximidade,
R   #State.Place_piece_on_conv; //uma mudança de estado ocorre;
S   #State.Wait_for_piece;
```

### NETWORK

TITLE = Estado: Wait\_for\_piece

//Espera por uma nova peça bruta. A chegada de uma nova peça é indicada pelo  
//sensor de proximidade da correia

Wait: AN #State.Wait\_for\_piece;

```
JC  TpCo;
R   #LED;                  //o LED é desligado;
A   #Proxy_switch;         //uma nova peça bruta chega,
R   #Transp_req;           //a correia é parada,
R   #State.Wait_for_piece; //e uma mudança de estado ocorre;
S   #State.Take_piece_from_conv;
```

### NETWORK

TITLE = Estado: Take\_piece\_from\_conv

//Neste estado a nova peça bruta é pega da correia para  
//a estação de trabalho

TpCo: AN #State.Take\_piece\_from\_conv;

```
JC  END;
A   #Clock_bit_q;          //o LED pisca rapidamente
=   #LED;                  //
AN  #Proxy_switch;         //quando a peça bruta é pega da correia,
R   #Conv_busy;           //a C. Transp. fica livre,
R   #LED;                 //o LED é desligado
R   #State.Take_piece_from_conv; //e uma mudança de estado ocorre;
S   #State.Process_piece;
```

END: BEU ;

END\_FUNCTION\_BLOCK

## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB2, parte 3)

```
FUNCTION_BLOCK "Transport"
TITLE = Controlando a Correia Transportadora
VERSION : 0.1

VAR_INPUT
    Initial : BOOL ;
    L_Barrier : BOOL ;
    Acknowledge : BOOL ;
    Transp_req : BOOL ;
    Clock_Bit : BOOL ;
END_VAR
VAR_OUTPUT
    LED : BOOL ;
    Conv_right : BOOL ;
    Conv_left : BOOL ;
    Count_Value :INT ;
END_VAR
VAR
    State : STRUCT
        Waiting : BOOL ;
        Conv_right : BOOL ;
        Assembly : BOOL ;
        Conv_left : BOOL ;
    END_STRUCT ;
    Count: "CTU";           // SFB 0 deve ser incluso
END_VAR
BEGIN
NETWORK
TITLE = Inicialização

    A   #Initial;
    S   #State.Waiting;
    R   #State.Conv_right;
    R   #State.Assembly;
    R   #State.Conv_left;
    CALL #Count (R:= #Initial,
                  CV := #Count_Value);

NETWORK
TITLE = Estado: "Waiting"
//A C. Transp. espera pela peça terminada neste estado.
    AN  #State.Waiting;
    JC  RIGH;
    R   #Conv_right;
    R   #Conv_left;
    R   #LED;
    A   #Transp_req;
    R   #State.Waiting;
    S   #State.Conv_right;

// (Continua na próxima página)
```

## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB2, parte 4)

```
NETWORK
TITLE = Estado: Conv_right
//Este estado descreve o transporte de peças terminadas na direção
//da montagem final
RIGH: AN  #State.Conv_right;
        JC  FINM;
        S   #Conv_right;
        A   #Clock_Bit;
        =   #LED;
        AN  #L_Barrier;
        R   #Conv_right;
        R   #State.Conv_right;
        S   #State.Assembly;
        AN  #L_Barrier;
        =   #L_Barrier;
        CALL #Count (CU      := #L_Barrier,
                     CV      := #Count_Value);

NETWORK
TITLE = Estado: Assembly
//Neste estado, a peça terminada é removida e uma nova peça bruta é deixada
//na correia. Após isto, o transporte da peça bruta na direção da estação
//de processamento desocupada parte com S4.
//
FINM: AN  #State.Assembly;
        JC  LEFT;
        S   #LED;
        A   #Acknowledge;
        R   #LED;
        R   #State.Assembly;
        S   #State.Conv_left;

NETWORK
TITLE = Estado: Conv_left
//Neste estado, o transporte da peça bruta para estação ocorre, que entrega
//a peça terminada.
LEFT: AN  #State.Conv_left;
        JC  ENDE;
        S   #Conv_left;
        A   #Clock_Bit;
        =   #LED;
        AN  #Transp_req;
        R   #Conv_left;
        R   #State.Conv_left;
        S   #State.Waiting;
ENDE: BEU ;

END_FUNCTION_BLOCK
```

## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB10, parte 5)

```
FUNCTION_BLOCK FB 10
TITLE =
VERSION : 0.1

VAR
  Station_1 : "Station";
  Station_2 : "Station";
  Station_3 : "Station";
  Transport : "Transport";
  Conv_busy : BOOL ;
END_VAR
VAR_TEMP
  Trans_1 : BOOL ;
  Trans_2 : BOOL ;
  Trans_3 : BOOL ;
  Trans : BOOL ;
END_VAR
BEGIN
NETWORK
TITLE =Station_1

  CALL #Station_1 (
    Initial           := "INITIALIZATION",
    Proxy_Switch      := "INI1",
    Acknowledge       := "S1",
    Clock_Bit_q       := "CLOCK_BIT_FAST",
    Clock_Bit_s       := "CLOCK_BIT_SLOW",
    LED               := "H1",
    Transp_req        := #Trans_1,
    Conv_busy         := #Conv_busy);

NETWORK
TITLE =Station_2

  CALL #Station_2 (
    Initial           := "INITIALIZATION",
    Proxy_Switch      := "INI2",
    Acknowledge       := "S2",
    Clock_Bit_q       := "CLOCK_BIT_FAST",
    Clock_Bit_s       := "CLOCK_BIT_SLOW",
    LED               := "H2",
    Transp_req        := #Trans_2,
    Conv_busy         := #Conv_busy);

NETWORK
TITLE =Station_3

  CALL #Station_3 (
    Initial           := "INITIALIZATION",
    Proxy_Switch      := "INI3",
    Acknowledge       := "S3",
    Clock_Bit_q       := "CLOCK_BIT_FAST",
    Clock_Bit_s       := "CLOCK_BIT_SLOW",
    LED               := "H3",
    Transp_req        := #Trans_3,
    Conv_busy         := #Conv_busy);
// (Continua na próxima página)
```

## Solução do Exercício 9.2: Contagem de Peças Terminadas (FB10, parte 6)

```

NETWORK
TITLE = Lógica: Transp_req
//Formação lógica para #Transp_req
O   #Trans_1;
O   #Trans_2;
O   #Trans_3;
=   #Trans;

NETWORK
TITLE = Transport

    CALL #Transport (
        Initial           := "INITIALIZATION",
        L_Barrier         := "LB1",
        Acknowledge       := "S4",
        Transp_req        := #Trans,
        Clock_Bit         := "CLOCK_BIT_FAST",
        LED               := "H4",
        Conv_right        := "K1_CONVR",
        Conv_left         := "K2_CONVL");

    L           #Transport.Count_Value ;
    ITD ;
    DTB ;
    T           QW12 ;
                                // Expande para DINT
                                // Converte para BCD

END_FUNCTION_BLOCK
DATA_BLOCK "ASSEMBLY_LINE_DB"
VERSION : 0.1
"ASSEMBLY_LINE"
BEGIN
END_DATA_BLOCK

ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1

VAR_TEMP
OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)
OB1_SCAN_1 : BYTE ;       //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)
OB1_PRIORITY : BYTE ;     //1 (Prioridade do 1 é baixa)
OB1_OB_NUMBR : BYTE ;     //1 (Bloco de Organização 1, OB1)
OB1_RESERVED_1 : BYTE ;   //Reservado para o sistema
OB1_RESERVED_2 : BYTE ;   // Reservado para o sistema
OB1_PREV_CYCLE : INT ;    //Ciclo de tempo da varredura anterior do OB1(milisegundos)
OB1_MIN_CYCLE : INT ;     //Mínimo cycle de tempo do OB1 (milisegundos)
OB1_MAX_CYCLE : INT ;     //Máximo cycle de tempo do OB1 (milisegundos)
OB1_DATE_TIME : DATE_AND_TIME ; //Data e horário da partida do OB1
END_VAR

BEGIN
NETWORK
TITLE =

    CALL FB  10 , DB  10 ;

END_ORGANIZATION_BLOCK

```

## Solução do Exercício 10.2: Comunicação com os SFBs GET/PUT (parte 1)

```
// Transfere os blocos compilados para a S7-400

DATA_BLOCK DB 14
VERSION : 0.1

"GET"
BEGIN
END_DATA_BLOCK

DATA_BLOCK DB 15
VERSION : 0.1

"PUT"
BEGIN
END_DATA_BLOCK

ORGANIZATION_BLOCK OB 1
TITLE = S7400 escreve no S7-300 e lê do S7-300
AUTHOR : PT41
FAMILY : A2_0
NAME : ST7PRO2
VERSION : 0.0

VAR_TEMP
OB1_EV_CLASS : BYTE ; //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)
OB1_SCAN_1 : BYTE ; //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)
OB1_PRIORITY : BYTE ; //1 (Prioridade do 1 é baixa)
OB1_OB_NUMBR : BYTE ; //1 (Bloco de Organização 1, OB1)
OB1_RESERVED_1 : BYTE ; //Reservado para o sistema
OB1_RESERVED_2 : BYTE ; // Reservado para o sistema
OB1_PREV_CYCLE : INT ; //Ciclo de tempo da varredura anterior do OB1(milisegundos)
OB1_MIN_CYCLE : INT ; //Mínimo cycle de tempo do OB1 (milisegundos)
OB1_MAX_CYCLE : INT ; //Máximo cycle de tempo do OB1 (milisegundos)
OB1_DATE_TIME : DATE_AND_TIME ; //Data e horário da partida do OB1
NDR_FLAG_14 : BOOL ;
ERROR_FLAG_14 : BOOL ;
DONE_FLAG_15 : BOOL ;
ERROR_FLAG_15 : BOOL ;
STATUS_WORD_14 : WORD ;
STATUS_WORD_15 : WORD ;
END_VAR

BEGIN
NETWORK
TITLE ="SFB_GET"
CALL SFB 14, DB 14 (
    REQ := I 28.0,
    ID := W#16#1,
    NDR := #NDR_FLAG_14,
    ERROR := #ERROR_FLAG_14,
    STATUS := #STATUS_WORD_14,
    ADDR_1 := P#I 0.0 BYTE 1,
    ADDR_2 := P#I 4.0 WORD 1,
    RD_1 := P#Q 40.0 BYTE 1,
    RD_2 := P#Q 42.0 WORD 1);
// (Continua na próxima página)
```

## Solução do Exercício 10.2: Comunicação com os SFBs GET/PUT (parte 2)

```
NETWORK
TITLE ="SFB_PUT"
CALL SFB 15, DB 15 (
    REQ      := I 28.1,
    ID       := W#16#1,
    DONE     := #DONE_FLAG_15,
    ERROR    := #ERROR_FLAG_15,
    STATUS   := #STATUS_WORD_15,
    ADDR_1   := P#Q 12.0 WORD 1,
    SD_1     := P#I 30.0 WORD 1);
```

```
NETWORK
TITLE =STATUS_WORD para QW38
A( ;
O  #NDR_FLAG_14;
O  #ERROR_FLAG_14;
) ;
JCN_002;
L  #STATUS_WORD_14;
T  QW 38;
_002: NOP 0;
```

```
NETWORK
TITLE =
A( ;
O  #DONE_FLAG_15;
O  #ERROR_FLAG_15;
) ;
JCN_001;
L  #STATUS_WORD_15;
T  QW 38;
_001: NOP 0;
```

```
NETWORK
TITLE =
// Caso contrário FFFF para QW38
A  I 28.0;
BEC ;
A  I 28.1;
BEC ;
L  W#16#FFFF;
T  QW 38;
END_ORGANIZATION_BLOCK
```



## Solução do Exercício 10.3: Comunicação com os SFBs START/STOP (parte 1)

// Transfere os blocos compilados para o S7-400

```
DATA_BLOCK DB 19
VERSION : 0.1
"START"
BEGIN
END_DATA_BLOCK
```

```
DATA_BLOCK DB 20
VERSION : 0.1
"STOP"
BEGIN
END_DATA_BLOCK
```

```
ORGANIZATION_BLOCK OB 1
TITLE =
VERSION : 0.1
VAR_TEMP
  OB1_EV_CLASS : BYTE ;      //Bits 0-3 = 1 (evento ocorrido), Bits 4-7 = 1 (evento Classe 1)
  OB1_SCAN_1 : BYTE ;        //1 (Cold restart 1o. ciclo do OB 1), 3 (ciclos 2 a n do OB 1)
  OB1_PRIORITY : BYTE ;      //1 (Prioridade do 1 é baixa)
  OB1_OB_NUMBR : BYTE ;      //1 (Bloco de Organização 1, OB1)
  OB1_RESERVED_1 : BYTE ;    //Reservado para o sistema
  OB1_RESERVED_2 : BYTE ;    // Reservado para o sistema
  OB1_PREV_CYCLE : INT ;     //Ciclo de tempo da varredura anterior do OB1(milisegundos)
  OB1_MIN_CYCLE : INT ;      //Mínimo cycle de tempo do OB1 (milisegundos)
  OB1_MAX_CYCLE : INT ;      //Máximo cycle de tempo do OB1 (milisegundos)
  OB1_DATE_TIME : DATE_AND_TIME ; //Data e horário da partida do OB1
```

```
DONE_FLAG_20 : BOOL ;
ERROR_FLAG_20 : BOOL ;
DONE_FLAG_19 : BOOL ;
ERROR_FLAG_19 : BOOL ;
STATUS_WORD_20 : WORD ;
STATUS_WORD_19 : WORD ;
END_VAR
```

```
BEGIN
NETWORK
TITLE =
//Entre com os caracteres "P_PROGRAM" em PI_NAME
  L 'P_PR';
  T MD 100;
  L 'OGRA';
  T MD 104;
  L 'M';
  T MB 108;
```

```
NETWORK
TITLE ="SFB_STOP"
  CALL SFB 20, DB 20 (
    REQ          := I 28.0,
    DONE          := #DONE_FLAG_20,
    ERROR         := #ERROR_FLAG_20,
    STATUS        := #STATUS_WORD_20);
```

// (Continua na próxima página)

## Solução do Exercício 10.3: Comunicação com os SFBs START/STOP (parte 2)

```
NETWORK
TITLE ="SFB_START"
CALL SFB 19, DB 19 (
    REQ      := I 28.1,
    DONE     := #DONE_FLAG_19,
    ERROR    := #ERROR_FLAG_19,
    STATUS   := #STATUS_WORD_19);
```

```
NETWORK
TITLE = STATUS_WORD para QW38
A( ;
O #DONE_FLAG_19;
O #ERROR_FLAG_19;
) ;
JCN_001;
L #STATUS_WORD_19;
T QW 38;
_001: NOP 0;
```

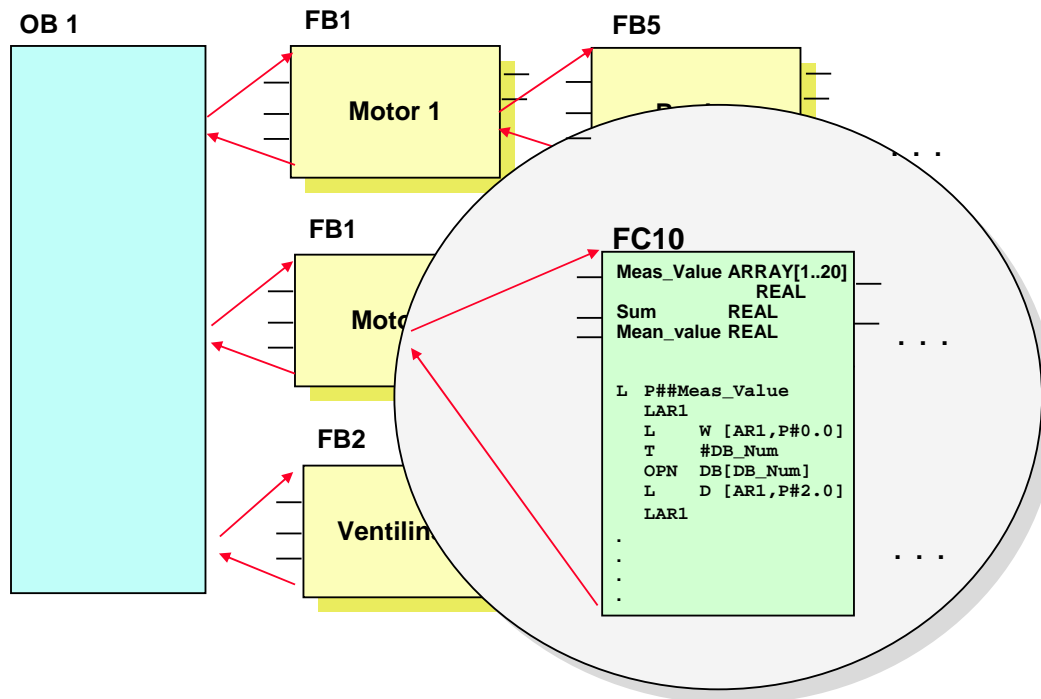
```
NETWORK
TITLE = STATUS_WORD para QW38
A( ;
O #DONE_FLAG_20;
O #ERROR_FLAG_20;
) ;
JCN_002;
L #STATUS_WORD_20;
T QW 38;
_002: NOP 0;
```

```
NETWORK
TITLE =

    A I 28.2;      // Caso contrário FFFF para QW38
    BEC ;
    A I 28.3;
    BEC ;
    L W#16#FFFF;
    T QW 38;
```

```
END_ORGANIZATION_BLOCK
```

## Apêndice: Acesso Indireto a Parâmetros dos FCs e FBs



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.1Conhecimento em Automação  
Trainings Center

### Conteúdo

	Pág.
Chamada de Funções com Tipos de Dados Complexos .....	2
Passando Parâmetros para Tipos de Dados Complexos .....	3
Acesso Indireto para Tipos de Dados Complexos .....	4
Passando Parâmetros para Ponteiros .....	5
Passando Parâmetros para Tipos de Parâmetros .....	6
Construção Especial para Parâmetros Atuais Elementares em DBs e Constantes .....	7
Exercício A.1: Avaliação do Parâmetro Data e Horário em uma FC .....	8
Chamada de FB Call com Tipos de Dados Complexos .....	9
Acesso Indireto a Parâmetros de Entrada/Saída .....	10
Acesso Indireto a Parâmetros de Entrada/Saída .....	11
"Passando" Parâmetros .....	12
Exercício A.2: Avaliação de Parâmetro Data e Horário em um FB .....	13
Exercício A.3: Avaliação de Parâmetros de Entrada/Saída em um FB .....	14
Solução do Exercício A.1: Acesso a Parâmetros DT em uma FC .....	15
Solução do Exercício A.2: Acesso a Parâmetros DT em uma FB .....	16
Solução do Exercício A.3: Acesso a Parâmetros Entrada/Saída em um FB (Parte 1) .....	17
Solução do Exercício A.3: Acesso a Parâmetros Entrada/Saída em um FB (Parte 2) .....	18

## Chamada de Funções com Tipos de Dados Complexos

### Exemplo: Passando um ARRAY para uma Função

#### A atribuição de parâmetros somente é possível simbolicamente

Network 1: Meas\_Val é declarado como um array na FC21

```
CALL FC 21
Meas_Val := "Temperature".sequence
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.2



Conhecimento em Automação  
Trainings Center

### Vista Geral

Parâmetros do tipo de dados complexos (ARRAY e STRUCT) oferecem um claro e eficiente caminho para transferência de grandes quantidades de dados relacionados entre o chamado e o bloco chamado e que pode se entender pelo conceito de "Programação Estruturada".

Um array ou uma estrutura pode ser passada para uma chamada de função como uma variável completa.

### Atribuindo Parâmetros

Para a passagem, um parâmetro do mesmo tipo de dado como o parâmetro atual a ser passado deve ser declarado na função chamada. Como um parâmetro (tipo de dado: ARRAY, STRUCT, DATE\_AND\_TIME e STRING) somente pode ser atribuído simbolicamente.

Desde que variáveis do tipo de dados complexos somente podem ser configuradas em blocos de dados ou em pilhas de dados locais, o parâmetro atual deve deste modo ser locado em um bloco de dados (DB global ou DB instance) ou na pilha de dados local do bloco chamado.

Após o editor STL/LAD/FBD ter verificado a compatibilidade dos tipos de dados do parâmetro atual e parâmetros de bloco na passagem de parâmetros de uma FC, somente um ponteiro com o número do DB ou um ponteiro de área cruzada é passado para a FC chamada para o parâmetro atual.

Este POINTER é configurado no L Stack do bloco chamado (área V) através da macro CALL. Este POINTER é então da maior importância para o programador em particular, quando o parâmetro passado tem que ser acessado indiretamente (ver apêndice).

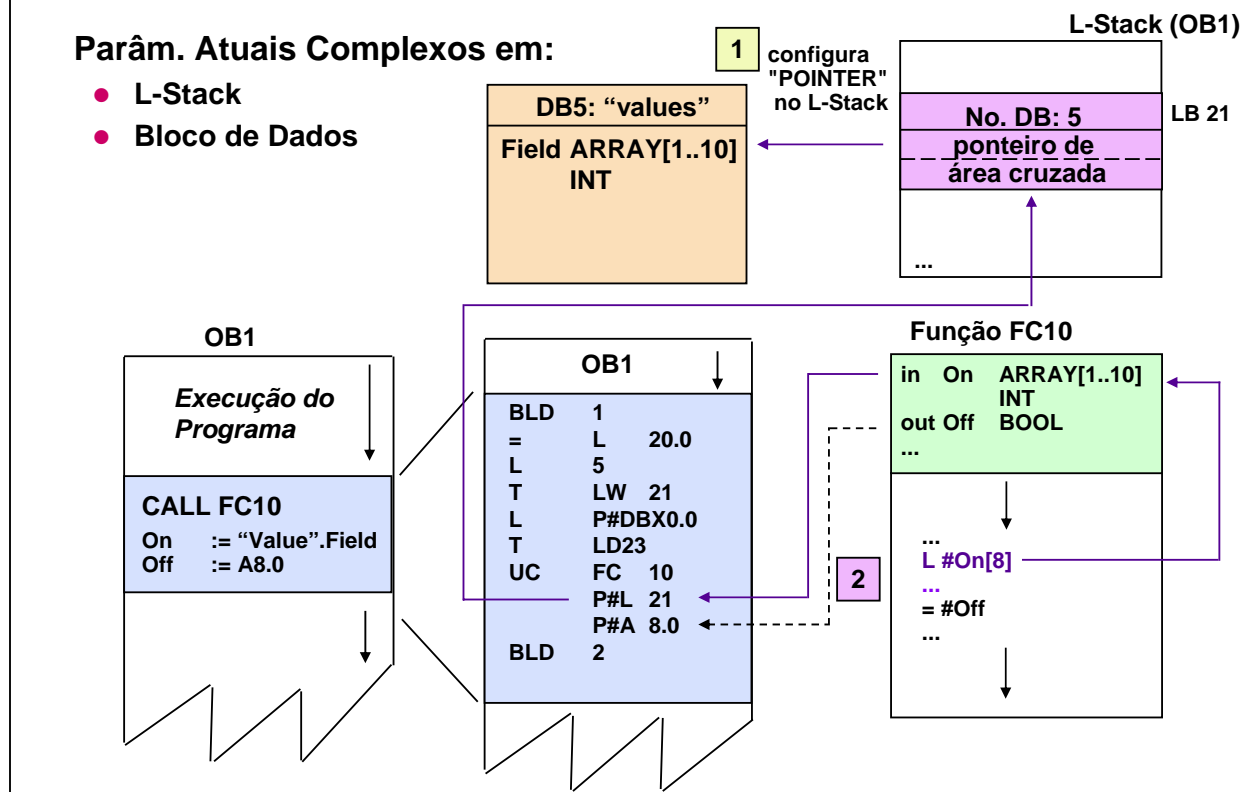
### Notas

- O número do dado local ocupado pode ser determinado pela seleção da opção de menu *View -> Block Properties*.
- Componentes dos ARRAYS ou STRUCTs podem também serem passados para um parâmetro de bloco se o parâmetro de bloco e os componentes ARRAY ou STRUCT são do mesmo tipo de dados.

## Passando Parâmetros para Tipos de Dados Complexos

Parâm. Atuais Complexos em:

- L-Stack
- Bloco de Dados



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PROJ\_15P.3



Conhecimento em Automação  
Trainings Center

### Passagem de Parâmetros

Com tipos de dados complexos (DT, STRING, ARRAY, STRUCT e UDT) os parâmetros atuais dinamizam em um bloco de dados ou na pilha de dados Locais (L-Stack) do bloco chamado (área V).

Uma vez que um ponteiro de área cruzada de 32 bits não pode achar um parâmetro atual em um DB, o Editor STL/LAD/FBD armazena na pilha local (L-Stack) do bloco chamado um "POINTER" de 48 bits, que aponta para o parâmetro atual.

Durante uma chamada, um ponteiro de área cruzada de 32 bits é passado para o "POINTER". Com a FC um acesso de parâmetros de parâmetros atuais então ocorre por meio de duplas aspas.

A configuração do "POINTER" na pilha L-Stack do bloco chamado ocorre antes do atual chavear para o FC chamado.

### Consequências

Os parâmetros do tipo de dado complexo são mais poderosos de operar do que tipos de parâmetros elementares. Parâmetros de entrada do tipo de dados complexos podem ser escritos no FC chamado sem passos adicionais.

Deste modo, parâmetros de saída podem ser processados sem passos adicionais.

## Acesso Indireto para Tipos de Dados Complexos

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_Val	ARRAY[1..8]		
*4.0			REAL		
32.0	out	Sum	REAL		
36.0	out	Mean_Val	REAL		
	in_out				
0.0	temp	DB_Num	WORD		

Network 1: Determining the DB-No. and the start address

```

L    P## Meas_Val          // Carrega endereço do POINTER no ACCU1
LAR1                               // e de lá carrega no AR1;
L    W [AR1,P#0.0]         // Determina o número do DB
T    #DB_Num               // e carrega na variável temporária;
OPN  DB[DB_Num]            // Abre DB
L    D [AR1,P#2.0]         // Determina área pointer
LAR1                               // e carrega no AR1;

```

Network 2: Cálculo da soma (sum)

```

L    0.000000e+000        // 0 no ACCU1 (sum =0.0)
L    8                     // Contador para ACCU1; Sum=0 p/ ACCU2
BEGN: TAK                  // Sum p/ ACCU1, contador p/ ACCU2
ENT                               // Contador p/ ACCU3
L    D[AR1,P#0.0]         // Componentes do campo no ACCU1
+R                               // Sum no ACCU1, contador p/ ACCU2
+AR1 P#4.0;                 // Incrementa AR1 de 4 bytes
TAK                             // Loop de contagem no ACCU1, sum no ACCU2
LOOP BEGN;                  // Decrementa loop contagem, salta se necessário
T    #Sum                  // Transfere sum para #Sum

```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.4



Conhecimento em Automação  
Trainings Center

### Acesso Indireto

Quando passando tipos de dados complexos tais como ARRAYS e STRUCTs, a potencialidade total somente pode ser alcançada se o parâmetro passado é combinado com o acesso indireto com o bloco chamado.

Um acesso indireto para os parâmetros atuais passados do tipo de dados complexos é feito em dois passos:

1. Primeiro, um ponteiro de área cruzada para o POINTER que tenha sido passado na pilha de dados locais é determinado por meio da operação:  
L P##Meas\_Val no bloco de chamada.
2. Para o acesso atual para os parâmetros atuais, este é então necessário avaliar a informação no POINTER, o qual referencia o operando corrente atual, tal como:

```

L P##Meas_Val // retorna ponteiro de área cruzada para o POINTER
LAR1          // carrega ponteiro de área cruzada no registrador de
              // endereços
L W[AR1,P#0.0] // retorna o número do DB do parâmetro atual,se
              // estiver
              // armazenado em um DB, caso contrário 0
L D[AR1,P#2.0] // retorna ponteiro de área cruzada para o parâ.
              // atual

```

O resultado é então calculado do modo usual.

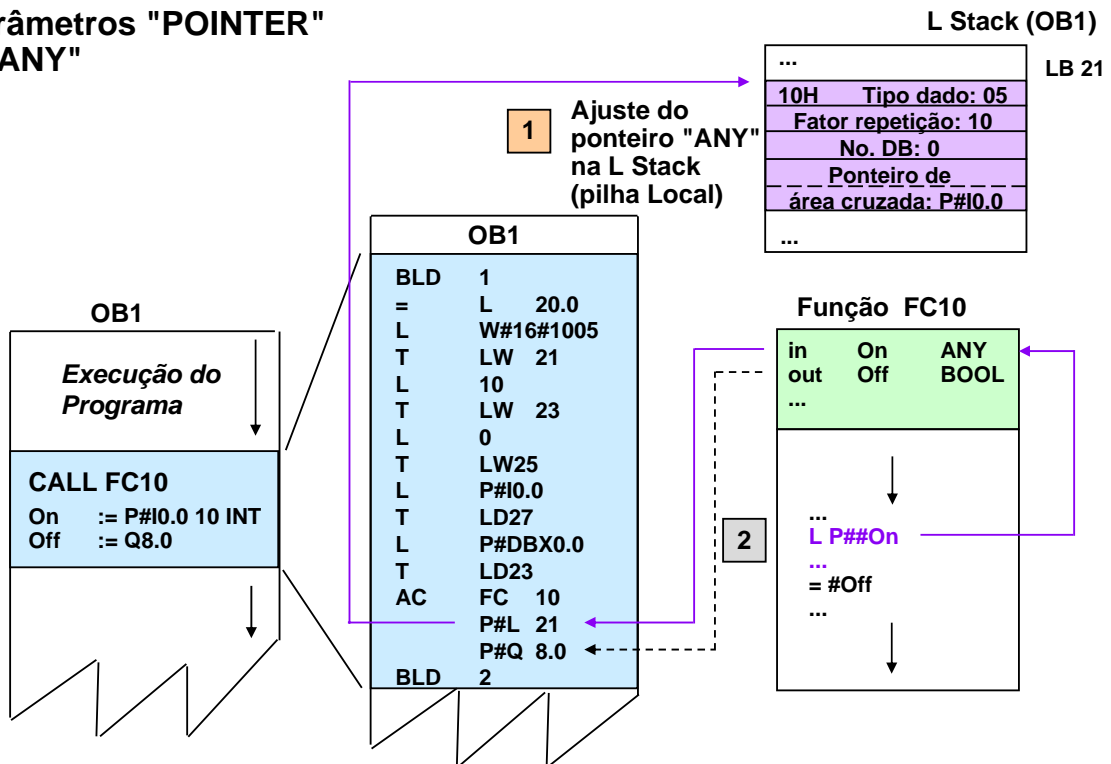
### Nota

Para obter acesso no exemplo acima, o programador deve ajustar os conteúdos do registrador de DB e o registrador AR1 de forma que o primeiro componente do campo transferido seja endereçado.

A instrução L Meas\_Val[0] também significa que o bloco de dados requisitado está aberto por meio do registrador de DBs e o registrador AR1 está ajustado para o início do ARRAY transferido.

## Passando Parâmetros para Ponteiros

Parâmetros "POINTER"  
e "ANY"



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.5Conhecimento em Automação  
Trainings Center

### Passagem de Parâmetros

Se um parâmetro do tipo de dado "POINTER" ou "ANY" é passado para uma FC, então o editor STL/LAD/FBD ajusta a estrutura de dados correspondente na pilha Local (L-Stack) do bloco chamado.

Com a FC chamada, um ponteiro de área cruzada 32 bits que aponta para esta estrutura de dados ("POINTER" ou "ANY") é então passada para a FC chamada.

Dentro da FC chamada não é possível acessar os parâmetros diretamente devido a perda do tipo de informação que são referenciadas através destes ponteiros "POINTER" ou "ANY".

A avaliação do conteúdo do "POINTER" ou "ANY" deve ser executada pelo usuário com comandos elementares STL dentro da FC chamada (ver Capítulo 2).

O acentamento da estrutura "POINTER" ou "ANY" na pilha Local do bloco chamado ocorre antes da mudança para a FC chamada.

### Exceção

Uma exceção na regra acima é o editor STL/LAD/FBD, quando em um parâmetro de bloco do tipo de dado "ANY" um parâmetro atual adicional do tipo de dado "ANY" é ajustado o qual é salvo na pilha L do bloco chamado.

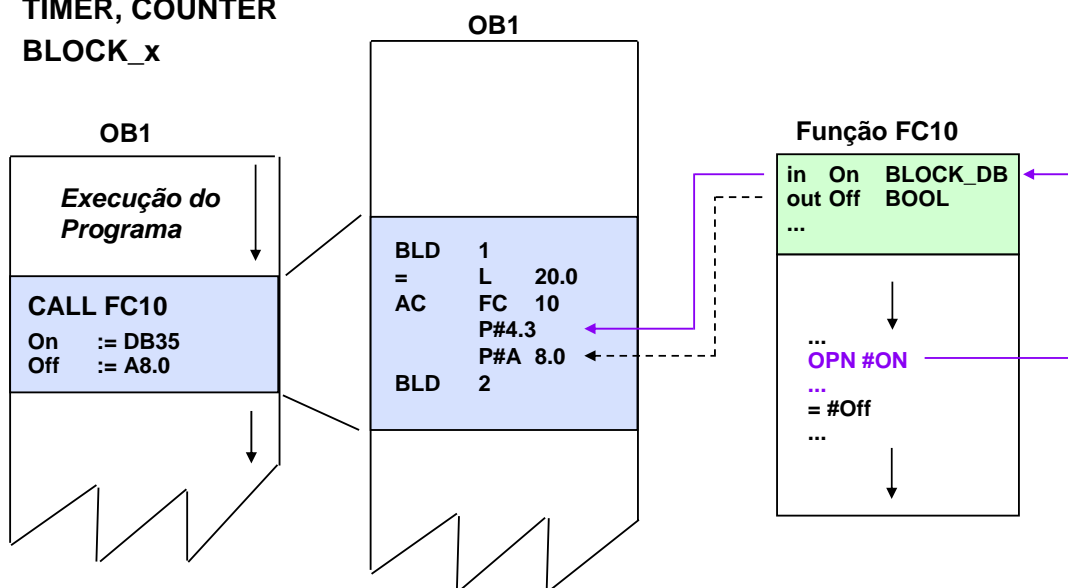
Neste caso o editor STL/LAD/FBD não seta um ponteiro adicional "ANY" na pilha Local do bloco que está chamando, mas passa diretamente para a FC um ponteiro de área cruzada para o ponteiro "ANY" já existente (na pilha L do bloco que chama).

Então, durante a execução, este ponteiro "ANY" pode ser manipulado pela chamada do bloco e então implementa uma atribuição "indireta" da FC com parâmetros atuais.

## Passando Parâmetros para Tipos de Parâmetros

### Parâmetros de Bloco:

- TIMER, COUNTER
- BLOCK\_x



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.6Conhecimento em Automação  
Trainings Center

### Passagem de Parâmetros

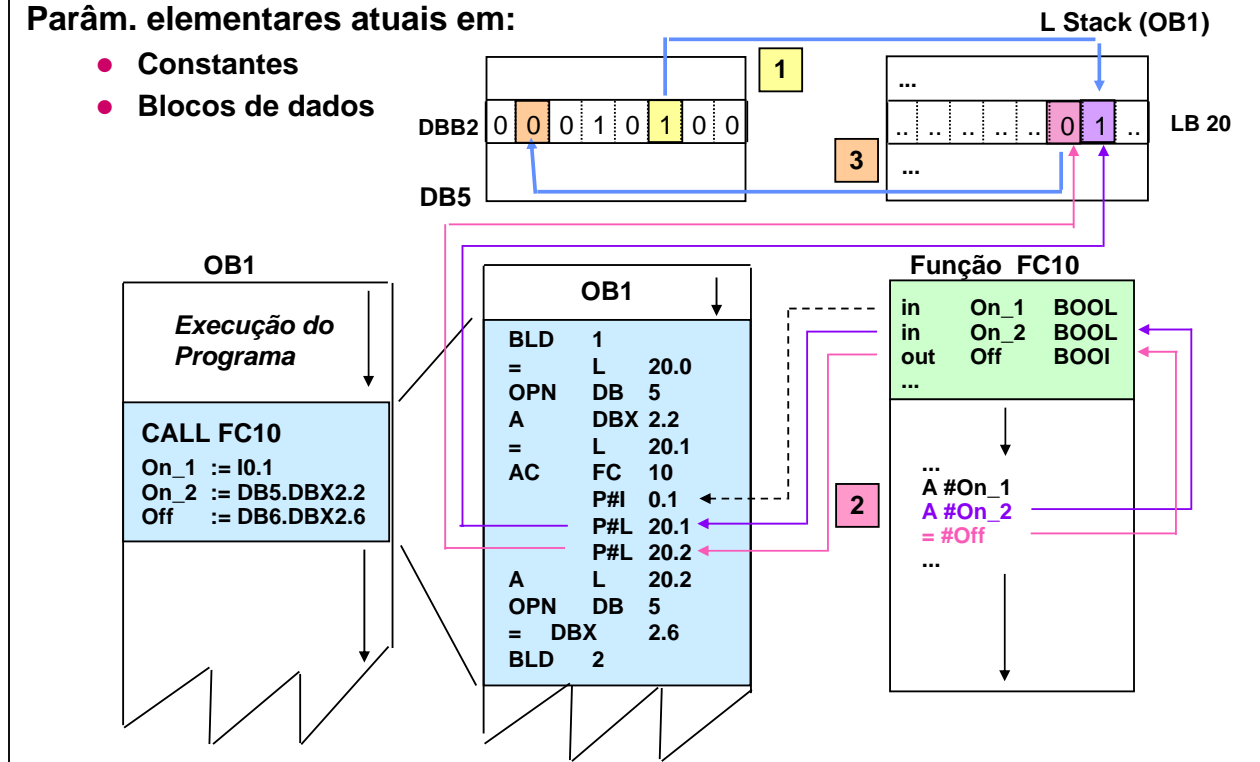
A passagem de parâmetros do tipo: TIMER, COUNTER e BLOCK\_x são fáceis. Neste caso, em vez de um ponteiro de área cruzada de 32 bits, o número do corrente TIMER ou COUNTER ou BLOCK\_xs é simplesmente passado para a FC chamada.



## Construção Especial para Parâmetros Atuais Elementares em DBs e Constantes

Parâm. elementares atuais em:

- Constantes
- Blocos de dados



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.7



Conhecimento em Automação  
Trainings Center

### Passem de Parâmetros

Se um parâmetro de entrada, saída ou entrada/saída é atribuído com uma constante ou com um parâmetro, que está guardado em um DB, então o editor STL/LAD/FBD primeiro de tudo reserva a memória necessária na pilha L do bloco que chama e então copia (com o parâmetro de entrada e entrada/saída) o valor do parâmetro atual na pilha local.

Para o parâmetro de saída, uma reserva de área de memória na pilha local ocorre mas sem inicialização.

Somente depois disto, ocorre a troca do atual na FC chamada, deste modo o editor STL/LAD/FBD passa em cada caso um ponteiro de área cruzada para a área de memória na pilha local da FC chamada.

Após um salto para trás no bloco chamado, o resultado (com parâmetros de saída e entrada/saída) será copiado nos parâmetros atuais.

### Consequências

Este mecanismo mostra que dentro de uma FC chamada, parâmetros de entrada somente podem ter verificação de estado e parâmetros de saída somente podem ser escritos.

Se um parâmetro de entrada é escrito, então através do valor correspondente é armazenado na pilha local, após o processamento da FC nenhum dado é transferido para os parâmetros atuais.

Do mesmo modo, parâmetros de saída somente podem ser escritos e não lidos. Com a leitura de um parâmetro de saída um valor indefinido é lido da pilha local devido a perda da inicialização.

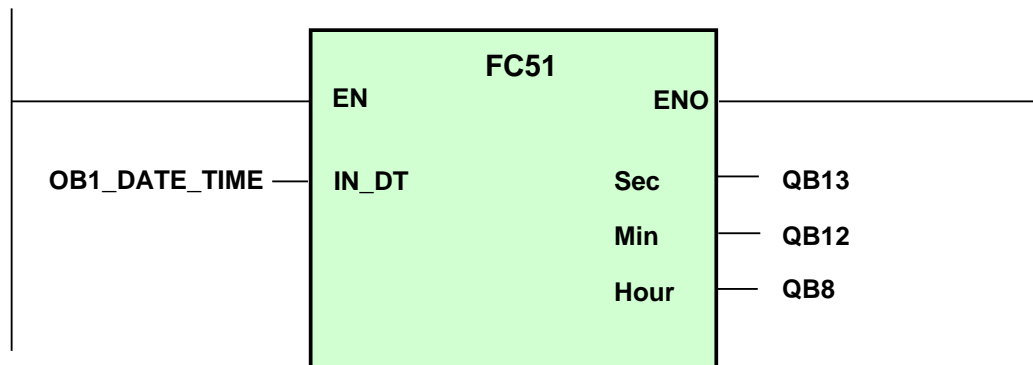
Parâmetros in/out causam estes problemas. Eles são atribuídos com os valores de parâmetros atuais antes da chamada bem como após a chamada.

### Importante

Parâmetros de saída devem se escritos em uma FC chamada (através de instruções como "S" e "R"), pelo contrário um valor indefinido da pilha local será copiado no parâmetro atual.

Se você não pode se certificar que parâmetros de saída serão escritos, você terá de usar parâmetros in/out em seu lugar.

## Exercício A.1: Avaliação do Parâmetro Data e Horário em uma FC



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.8Conhecimento em Automação  
Trainings Center

### Vista Geral

O seguinte exercício deverá demonstrar como você pode acessar indiretamente parâmetros de entrada, saída e transientes dos tipos de dados complexos dentro de uma função.

Você deve usar a mesma tecnologia se você tiver que acessar indiretamente outros tipos de dados complexos, tais como ARRAYS, STRUCTs ou STRINGs.

### Definição da Tarefa

Criar uma FC51 com as seguintes propriedades:

- A FC51 tem um parâmetro de entrada #IN\_DT do tipo de dado: DATE\_AND\_TIME
- Nestes 3 parâmetros de saída #Sec, #Min e #Hour, a FC51 retorna componentes segundos, minutos e horas do parâmetro DT passado a ele.

### Execução

1. Criar uma FC51 com as propriedades necessárias.
2. Chamar a FC51 no OB1. Alimentar o parâmetro de bloco #IN\_DT com a variável OB1\_DATE\_TIME das informações de partida do OB1.
3. Carregar os blocos para a CPU e testar o programa.

## Chamada de FB Call com Tipos de Dados Complexos

### Exemplo: Passando ARRAYS para um Bloco de Funções

**FB17**

Address	Decl.	Name	Type	Initial Value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0	in		REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT		
		temp			

**DB2 "Temperature"**

Address	Name	Type	Initial
0.0		STRUCT	
+0.0	Cylinder	ARRAY[1..10]	
*4.0		REAL	
+40.0	Shaft	ARRAY[1..15]	
*4.0		REAL	
=100.0		END_STRUCT	

#### Atribuindo parâmetros complexos somente é possível simbolicamente

Network 1:

```
CALL FB 17, DB 30
Meas_1 := "Temperature".Cylinder
Sum_1 := MD20
Sum_2 := MD30
Meas_2 := "Temperature".Shaft
```

## SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.9



Conhecimento em Automação  
Trainings Center

### Tipos de Dados Complexos

Do mesmo modo que com as funções, endereçamento de tipos de dados complexos (ARRAY, STRUCT, DATE\_AND\_TIME e STRING) podem ser passados completamente para um bloco de funções chamado.

Para a passagem, um parâmetro do mesmo tipo de dado como um parâmetro atual a ser passado deve ser declarado no bloco de funções chamado.

A atribuição como um parâmetro somente é possível simbolicamente.

### Parâmetros de Entrada e Saída

Para parâmetros de entrada e saída do tipo de dados complexos, áreas correspondentes para os valores dos parâmetros atuais são acentadas no DB instance. Quando chamando o FP, os parâmetros atuais dos parâmetros de entrada são então copiados no DB instance usando a SFC 20 (BLKMOV) ("passando pelo valor"), antes o atual chaveia para a seção de instruções do FP.

Do mesmo modo, os valores dos parâmetros de saída são copiados do DB instance de volta para os parâmetros atuais após o FP ter sido processado.

Como um resultado, uma quantidade não significativa de cópias (tempo do processamento) pode ocorrer na atribuição dos parâmetros de entrada e saída. Esta quantidade de cópias são baipassadas com parâmetros in/out.

### Parâmetros In/out

Nenhuma "passagem pelo valor" ocorre com parâmetros in/out do tipo de dados complexos. Seis bites são meramente reservados para cada parâmetro in/out na área de dados instance. Um POINTER para os parâmetros atuais é inserido nestes bites ("passagem por referência").

### Notas

Parâmetros de entrada e saída do tipo de dados complexos podem ser inicializados na seção de declarações de um FP, mas não como parâmetros in/out.

Parâmetros de entrada e saída do tipo de dados complexos não tem que ser atribuídos na chamada de um FB, parâmetros in/out tem que ser atribuídos.

O acesso a memória ou registro indireto a parâmetros entrada/saída ou parâmetros in/out do tipo de dados complexos é diferente dos parâmetros elementares.

## Acesso Indireto a Parâmetros de Entrada/Saída

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0			REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT	0	

Network 1: Determinando o endereço de início do Meas\_1

```

LAR1 P##Meas_1      // Carrega ponteiro de área cruzada p/parâmetro sem
                    // offset de endereço (multi-instances) no AR1
TAR2                // Carrega offset de endereço no ACCU1
+AR1                // Soma offset de endereço no AR1;
                    // AR1 agora aponta p/parâmetros no DB instance
                    // DB instance já está aberto

```

Network 2: Acesso para Meas\_1

```

L      0.000000e+000 // 0 no ACCU1 (Soma =0.0)
L      10            // Contador para ACCU1; Sum=0 p/ ACCU2
BEGN: TAK           // Sum no ACCU1, contador no ACCU2
      ENT           // Contador p/ ACCU3
      L      D[AR1,P#0.0] // Campo de componente no ACCU1
      +R           // Soma no ACCU1, contador p/ ACCU2
      +AR1 P#4.0;    // Incrementa AR1 em 4 bytes
      TAK           // Loop de contagem no ACCU1, soma no ACCU2
      LOOP BEGN;    // Decrementa loop de contagem e salta se necessário
      T      #Sum_1  // Transfere soma para #Sum_1

```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.10



Conhecimento em Automação  
Trainings Center

### Acesso Indireto

Quando passando tipos de dados complexos tais como ARRAYS e STRUCTs, a potência total somente pode ser explorada se o parâmetro passado é combinado com o acesso indireto dentro do bloco chamado.

Um acesso indireto ao parâmetro atual passado feito em dois passos:

1. Primeiro, por meio da operação:

```

LAR1 P##Meas_1 // retorna ponteiro de área cruzada p/parâmetro
                // sem offset de endereço

```

um ponteiro de área cruzada para os parâmetros no DB instance é carregado no AR1.

O ponteiro determinado deste modo contém o identificador de área ID e o mesmo endereço byte.bit que também é mostrado pelo editor durante a declaração de parâmetros na primeira coluna da parte de declarações.

No caso de um multi-instance, este não corresponde ao endereço atual do parâmetro de entrada/saída no DB instance. Este é também necessário para somar o offset de endereço do AR2, o qual identifica o início da área de dados instance no caso multi-instance, para o ponteiro no AR1.

```

TAR2                // Carrega offset de endereço no ACCU1
+AR1                // Soma offset de endereço p/ AR1;

```

2. Após isto, o acesso atual aos parâmetros entrada/saída podem ocorrer. O DB instance não deve ser aberto especialmente, como ele já tinha sido aberto pela macro CALL na chamada do FB.

```

L      D[AR1,P#0.0] // Carrega primeiro componente do Meas_1 etc.

```

### Nota

Acesso indireto aos parâmetros entrada/saída e in/out dos tipos de dados elementares ou para as variáveis estáticas é feito do mesmo modo, como neste caso também valores dos operandos são guardados no DB instance.

## Acesso Indireto a Parâmetros de Entrada/Saída

Address	Declaration	Name	Type	Start value	Comment
0.0	in	Meas_1	ARRAY[1..10]		
*4.0			REAL		
40.0	out	Sum_1	REAL	0.000000e+000	
44.0	out	Sum_2	REAL	0.000000e+000	
48.0	in_out	Meas_2	ARRAY[1..15]		
*4.0	in_out		REAL		
54.0	stat	DB_Num	INT	0	

Network 3: Determinando o endereço inicial do Meas\_2

```

LAR1 P##Messung_2    // Carrega ponteiro de área cruzada p/ POINTER sem
TAR2                // Carrega offset de endereço no ACCU1, soma ao AR1;
+AR1                // AR1 agora aponta p/ POINTER no DB instance
L    W [AR1,P#0.0]   // Carrega número do DB do POINTER no ACCU1
T    #DB_Num         // Transfere número do DB(ou 0) na variável estática
OPN DB [#DB_Num]     // Abre DB
L    D [AR1,P#2.0]   // Carrega ponteiro de área cruzada p/ parâmetro
LAR1                // Carrega ponteiro no AR1, AR1 aponta p/parâmetro

```

Network 4: Access to Meas\_2

```

L    0.000000e+000   // 0 p/ ACCU1 (Soma =0.0)
L    15              // Contador p/ ACCU1; Soma=0 p/ ACCU2
BEGN: TAK            // Soma no ACCU1, contador no ACCU2
ENT                // Contador p/ ACCU3
L    D[AR1,P#0.0]    // Campo de componentes no ACCU1
+R                // Soma no ACCU1, contador p/ ACCU2
...                // ...

```

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.11



Conhecimento em Automação  
Trainings Center

### Acesso Indireto

O acesso indireto aos parâmetros in/out do tipo de dados complexos diferem em estrutura comparados ao acesso aos parâmetros de entrada e saída.

Com parâmetros in/out do tipo de dados complexos, o valor não é copiado, mas somente um POINTER para o parâmetro in/out no DB instance.

O acesso atual ocorre em três passos:

1. Primeiro, por meio da operação:

```
LAR1 P## Meas_2 // retorna o ponteiro de área cruzada p/ POINTER
```

um ponteiro de área cruzada para o POINTER transferido é carregado no registrador AR1. Como no caso anterior, o endereço byte.bit do ponteiro no AR1 não identifica o endereço atual do POINTER no DB instance.

No caso do multi-instance o offset do AR2 deve ainda ser somado ao ponteiro do registrador AR1:

```
TAR2                // Carrega endereço no ACCU1, soma ao AR1;
+AR1                // AR1 agora aponta p/ POINTER no DB instance;
```

2. No próximo passo a informação no POINTER é avaliada, se necessário o DB, no qual o parâmetro atual está localizado, é aberto e um ponteiro de área cruzada para os operandos atuais é carregado no registrador AR1:

```

L    W [AR1,P#0.0] // Carrega número do DB do POINTER no
                // ACCU1
T    #DB_Num      // Transfere número do DB (ou 0) na variável
OPN DB [#DB_Num] // Abre DB
L    D [AR1,P#2.0] // Carrega ponteiro de área cruzada p/ parâmetro
LAR1                // Carrega ponteiro no AR1, AR1 aponta p/parâm.

```

3. Após isto, o acesso atual para o parâmetro atual pode ocorrer:

```
L D[AR1,P#0.0] // Carrega primeiro componente do Meas_2 etc.
```

## "Passando" Parâmetros

**Tamanho do aninhamento:**

- S7-300: máx. 8      S7-400: máx. 16



**A passagem depende do tipo de bloco, dado e parâmetro:**

Chamada	FC chama FC	FB chama FC	FC chama FB	FB chama FB
Tipo de dado	P E C	P E C	P E C	P E C
Input -> Input	X - -	X X -	X - X	X X X
Output -> Output	X - -	X X -	X - -	X X -
in/out -> Input	X - -	X - -	X - -	X - -
in/out -> Output	X - -	X - -	X - -	X - -
in/out -> in/out	X - -	X - -	X - -	X - -

E: Tipo de dado elementar

C: Tipo de dado complexo

P: Tipo parâmetro (Timer, Counter, Block\_x)

### SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.12



Conhecimento em Automação  
Trainings Center

### Vista Geral

A "passagem" de parâmetros de bloco é uma forma especial de acesso ou atribuição de parâmetros de bloco. "Passagem" significa que o parâmetro formal do bloco chamado recebe o parâmetro atual do bloco chamado.

### Restrições com Relação aos Tipos de Parâmetros

Como uma regra geral, o parâmetro atual deve ser do mesmo tipo de dado que o parâmetro formal. Mais adiante, parâmetros de entrada do bloco chamado somente podem ser ajustados em um parâmetro de entrada do bloco chamado e parâmetros de saída somente em parâmetros de saída. Um parâmetro in/out do bloco chamado pode em princípio ser ajustado como parâmetros de entrada, saída e in/out do bloco chamado.

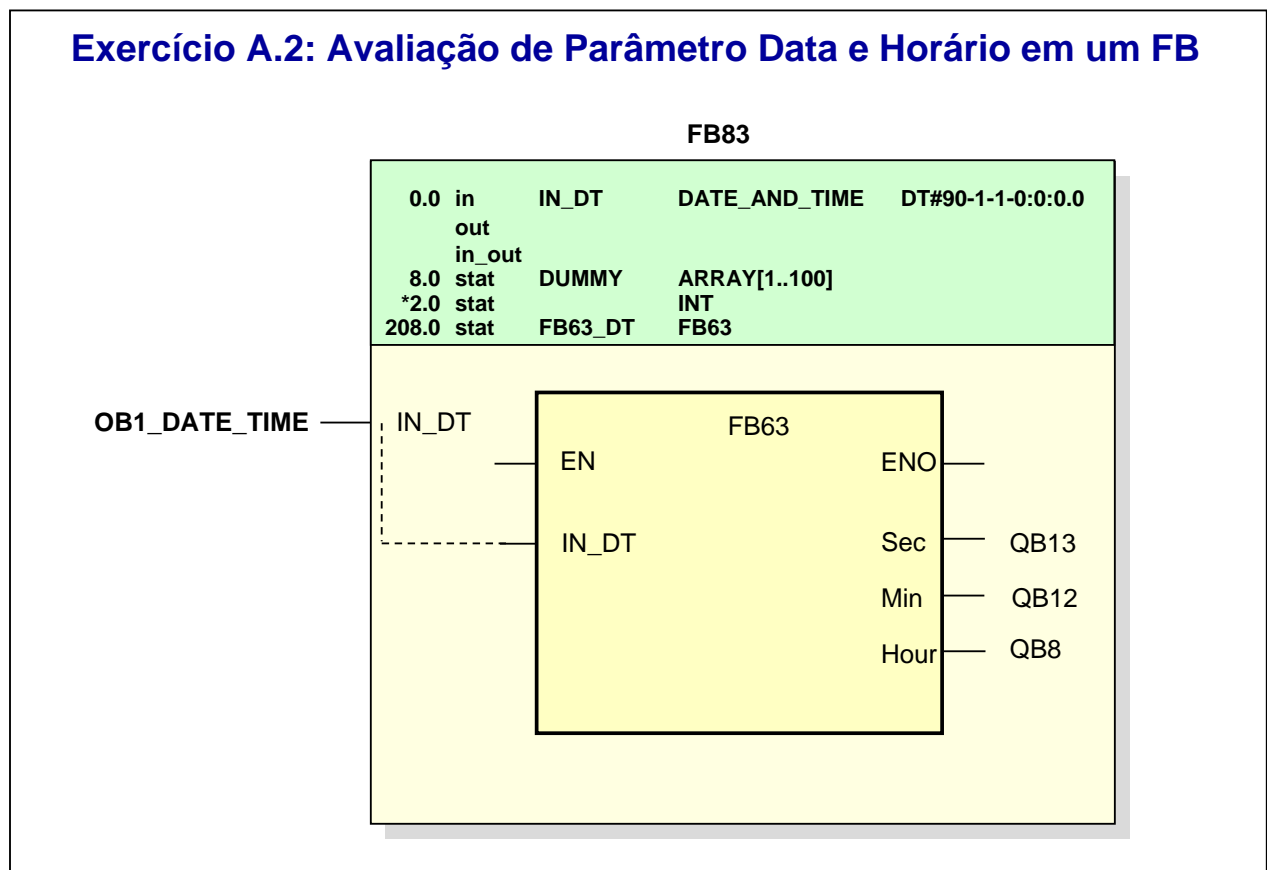
### Restrições com Relação aos Tipos de Dados

Com relação aos tipos de dados, existem restrições dependendo das diferentes armazenagens dos parâmetros do bloco na chamada de FC ou FB. Parâmetros de blocos do tipo de dado elementar podem ser passados sem restrições. Tipos de dados complexos de parâmetros de entrada e saída podem então ser passados se o bloco chamado é um FB. Parâmetros de blocos com os tipos de parâmetros: formato TIMER, COUNTER e BLOCK\_x então somente podem ser passados de um parâmetro de entrada para um parâmetro de entrada, se o bloco chamado é um FB.

### Nota

A "passagem" dos tipos de parâmetros: TIMER, COUNTER e BLOCK\_x podem ser implementados em FCs com a ajuda de endereçamento indireto. O número desejado de TIMER, COUNTER ou BLOCK\_x é, por exemplo, passado como um parâmetro do tipo de dado WORD do bloco que chama para o bloco chamado. Com a última chamada de bloco, este parâmetro pode então ser transferido para uma variável temporária e através desta variável o respectivo TIMER, COUNTER ou BLOCK\_x pode então ser processado.

## Exercício A.2: Avaliação de Parâmetro Data e Horário em um FB



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.13Conhecimento em Automação  
Trainings Center

### Vista Geral

O exercício seguinte deve demonstrar como você pode acessar indiretamente um parâmetro de entrada ou saída do tipo de dado complexo dentro de um bloco de funções com capacidade multi-instance.

Você deve usar a mesma tecnologia se você tem que acessar indiretamente outros tipos de dados complexos, tais como ARRAYS, STRUCTs ou STRINGs.

### Definição da Tarefa

Criar um FB63 com as seguintes propriedades:

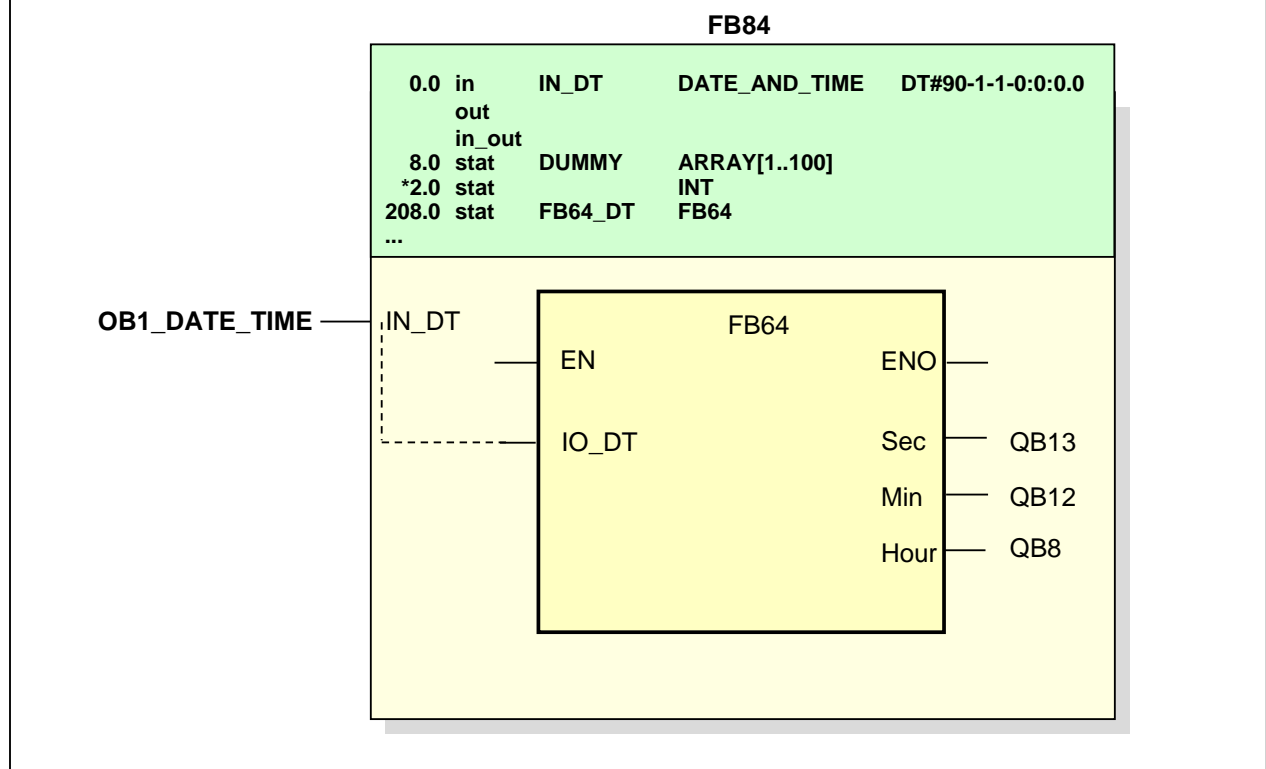
- O FB63 tem um parâmetro de entrada `#IN_DT` do tipo de dado: `DATE_AND_TIME`
- Em seus 3 parâmetros de saída `#Sec`, `#Min` e `#Hour`, o FB63 retorna os componentes segundos, minutos e horas do parâmetro DT passado a ele.

### Execução

1. Criar um FB63 com as propriedades requeridas.
2. Para testar se o FB63 gerado também está realmente com capacidade multi-instances chamar um instance do FB63 em um FB de alto nível.  
Criar um FB83 de alto nível. Primeiro, declare um parâmetro de entrada `#IN_DT` do tipo DT no FB83. Então declare uma variável estática `#DUMMY` do tipo `ARRAY[1..100] OF INT` e um instance do FB63 com o nome `#FB63_DT`.
3. Chame o instance `#FB63_DT` dentro do FB83 e alimente-o com o parâmetro de entrada `#IN_DT` deste instance com o parâmetro de entrada `#IN_DT` do FB83.  
Alimente o parâmetro de entrada do instance `FB63_DT` com os bytes de saída `QB8`, `QB12` e `QB13`.
4. Chame o FB83 com o instance `DB83` no `OB1`. Atribua o parâmetro de entrada `#IN_DT` com a variável `OB1_DATE_TIME` das informações iniciais do `OB1`.
5. Carregue os blocos para a CPU e teste seu programa.



## Exercício A.3: Avaliação de Parâmetros de Entrada/Saída em um FB



SIMATIC S7

Siemens AG 1999. All rights reserved.

Datum: 04.10.2007  
Datei: PRO2\_15P.14Conhecimento em Automação  
Trainings Center

### Vista Geral

O exercício seguinte deve demonstrar como você pode acessar indiretamente um parâmetro in/out do tipo de dado complexo dentro de um bloco de funções com capacidade multi-instance.

You must use the same technology if you have to indirectly access other complex data types, such as ARRAYS, STRUCTs or STRINGs.

### Definição de Tarefa

Criar um FB64 com as seguintes propriedades:

- O FB64 tem um parâmetro de in/out #IO\_DT do tipo de dado DT.
- Nos 3 parâmetros de saída #Sec, #Min e #Hour, o FB64 retorna os componentes segundos, minutos e horas do parâmetro DT passado a ele.

### Execução

1. Criar um FB64 com as propriedades requisitadas.
2. Criar um FB84. Primeiro, declare um parâmetro de entrada #IN\_DT do tipo DT no FB84. Então declare uma variável estática #DUMMY do tipo ARRAY[1..100] OF INT e um instance do FB64 com o nome #FB64\_DT.
3. Chame o instance #FB64\_DT dentro do FB84 e atribua o parâmetro in/out #IO\_DT deste instance com o parâmetro de entrada #IN\_DT do FB84. Note que a passagem direta do parâmetro de entrada para um parâmetro in/out não está permitido. Qual solução é recomendada.  
Alimente o parâmetro de saída do instance FB64\_DT com os bytes de saída QB8, QB12 e QB13, como na tarefa anterior.
4. Chame o FB84 com instance DB84 no OB1. Atribua o parâmetro de entrada #IN\_DT do FB84 com a variável OB1\_DATE\_TIME das informações iniciais do OB1.
5. Carregue os blocos para a CPU e teste o programa.



## Solução do Exercício A.1: Acesso aos parâmetros DT em uma FC

```
FUNCTION FC 51 : VOID
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR_OUTPUT
    Sec : BYTE ;
    Min : BYTE ;
    Hour : BYTE ;
END_VAR
VAR_TEMP
    DB_Num : WORD ;
END_VAR
BEGIN
NETWORK
TITLE =
// No caso de um parâmetro de entrada, saída ou in/out do tipo de dado complexo, um
// ponteiro de área cruzada para uma variável POINTER (6 bytes), que é ajustado na pilha
// local do bloco que chama pela macro CALL, é transferido p/ a função chamada pela
// passagem de parâmetro. O conteúdo da variável POINTER aponta para o operando atual.
// Para acesso indireto, um ponteiro de área cruzada para este POINTER é criado primeiro.
// No próximo estágio, o conteúdo da variável POINTER é lida e o acesso é feito para o
// operando atual através desta informação.
//
    L    P##IN_DT;           // Carrega ponteiro de área cruzada p/ POINTER no ACCU1
    LAR1 ;                   // Carrega ponteiro no AR1, AR1 agora aponta p/ POINTER
    L    W [AR1,P#0.0];      // Carrega número do DB do POINTER no ACCU1
    T    #DB_Num;           // Transfer número do DB (ou 0) na variável temporária
    OPN  DB [#DB_Num];      // Abre DB
    L    D [AR1,P#2.0];      // Carrega ponteiro de área cruzada p/ variável DT do
                                // POINTER
    LAR1 ;                   // Carrega ponteiro no AR1, AR1 agora aponta p/variável DT
    L    B [AR1,P#3.0];      // Carrega componente horas da variável DT
    T    #Hour;              // Transfere para parâmetro de saída #Hour
    L    B [AR1,P#4.0];      // Carrega componente minutos da variável DT
    T    #Min;               // Transfere para parâmetro de saída #Min
    L    B [AR1,P#5.0];      // Carrega componente segundos da variável DT
    T    #Sec;               // Transfere para parâmetro de saída #Sec
    SET  ;
    SAVE ;                   // Seta o bit BR em 1
END_FUNCTION

ORGANIZATION_BLOCK OB1
TITLE =
VERSION : 0.1
VAR_TEMP
    OB1_TEMP: ARRAY[1..20] OF BYTE; // Informações iniciais do OB1
BEGIN
NETWORK
TITLE =
    CALL FC 51 (
        IN_DT := #OB1_DATE_TIME,
        Sec   := QB 13,
        Min   := QB 12,
        Hour   := QB 8);
END_ORGANIZATION_BLOCK
```

## Solução do Exercício A.2: Acesso a Parâmetros DT em um FB

```

FUNCTION_BLOCK FB 63
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR_OUTPUT
    Sec : BYTE ;
    Min : BYTE ;
    Hour : BYTE ;
END_VAR
BEGIN
NETWORK
TITLE =
// No caso de uma entrada, o parâmetro de saída do tipo de dado complexo, o valor
// da variável complexa é copiada ou copiada de volta no DB instance.
// Para acesso indireto, um ponteiro de área cruzada incluindo o offset de endereço AR2,
// o qual ocorre no caso de um multi-instance, é criado primeiro.
//
    LAR1 P##IN_DT;      // Carrega ponteiro de área cruzada p/ #IN_DT sem o
                        // offset de endereço adicional no AR2
    TAR2 ;              // Transfere offset de endereço no ACCU1-offset endereço no
                        // AR2
                        // é feito pela macro CALL
    +AR1 ;              // Carrega conteúdo do ACCU1 (offset endereço AR2) p/AR1,
                        // AR1 agora aponta p/ #IN_DT
    L   B [AR1,P#3.0];  // Carrega componente horas da variável DT
    T   #Hour;          // Transfere p/ parâmetro de saída #Hour
    L   B [AR1,P#4.0];  // Carrega componente minutos da variável DT
    T   #Min;           // Transfere p/ parâmetro de saída #Min
    L   B [AR1,P#5.0];  // Carrega componente segundos da variável DT
    T   #Sec;           // Transfere p/ parâmetro de saída #Sec
    SET ;
    SAVE ; // Seta o bit BR em 1
END_FUNCTION_BLOCK

FUNCTION_BLOCK FB 83
TITLE =
VERSION : 0.1

VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR
    DUMMY : ARRAY [1 .. 100] OF INT ;
    FB63_DT : FB 63;
END_VAR
BEGIN
NETWORK
TITLE =

    CALL #FB63_DT (
        IN_DT      := #IN_DT,
        Sec        := QB 13,
        Min        := QB 12,
        Hour       := QB 8);

END_FUNCTION_BLOCK

```

### Solução do Exercício A.3: Acesso a Parâmetros Entrada/Saída em um FB (Parte 1)

FUNCTION\_BLOCK FB 64

TITLE =

VERSION : 0.1

VAR\_OUTPUT

Sec : BYTE ;

Min : BYTE ;

Hour : BYTE ;

END\_VAR

VAR\_IN\_OUT

IO\_DT : DATE\_AND\_TIME ;

END\_VAR

VAR\_TEMP

DB\_Num : WORD ;

END\_VAR

BEGIN

NETWORK

TITLE =

// No caso de um parâmetro in/out do tipo de dado complexo, o valor da variável complexa  
 // não é copiada no DB instance, mas ao contrário um POINTER para o operando atual  
 // é armazenado no DB instance. Para acesso indireto, um ponteiro de área cruzada  
 // para este POINTER, incluindo o offset de endereço AR2, o qual ocorre no caso de um  
 // multi-instance, é criado primeiro.  
 // Após isto, o acesso é feito para os operandos atuais no modo usual.

```

LAR1 P##IO_DT;    // Carrega ponteiro de área cruzada para o POINTER
                  // sem offset de endereço
TAR2 ;            // Transfere o offset de endereço do AR2 no ACCU1
+AR1 ;            // Soma o offset de endereço para AR1, AR1 agora aponta para
                  // o POINTER
L   W [AR1,P#0.0]; // Carrega número do DB do POINTER no ACCU1
T   #DB_Num;       // Transfere o número do DB (ou 0) em variável temporária
OPN DB [#DB_Num]; // Abre o DB
L   D [AR1,P#2.0]; // Carrega ponteiro de área cruzada para a variável DT      //
                  doPOINTER
LAR1 ;            // Carrega ponteiro no AR1, AR1 aponta para a variável DT
T   #Hour;         // Transfere para o parâmetro de saída #Hour
L   B [AR1,P#4.0]; // Carrega componente minutos da variável DT
T   #Min;          // Transfere para o parâmetro de saída #Min
L   B [AR1,P#5.0]; // Carrega componente segundos da variável DT
T   #Sec;          // Transfere para o parâmetro de saída #Sec
SET ;
SAVE ;            // Seta o bit BR em 1
  
```

END\_FUNCTION\_BLOCK

Solução do Exercício A.3: Acesso a Parâmetros Entrada/Saída em um FB (Parte 2)

```
FUNCTION_BLOCK FB 84
TITLE =
VERSION : 0.1
VAR_INPUT
    IN_DT : DATE_AND_TIME ;
END_VAR
VAR
    DUMMY : ARRAY [1 .. 100] OF INT ;
    FB64_DT : FB 64;
END_VAR
VAR_TEMP
    DT_TEMP : DATE_AND_TIME ;
    I_Ret_VAL : INT ;
END_VAR
BEGIN
NETWORK
TITLE =

    CALL SFC 20 (
        SRCBLK    := #IN_DT,
        RET_VAL    := #I_Ret_VAL,
        DSTBLK     := #DT_TEMP);

    CALL #FB64_DT (
        Sec        := AB    13,
        Min        := AB    12,
        Hour       := AB     8,
        IO_DT      := #DT_TEMP);

END_FUNCTION_BLOCK

ORGANIZATION_BLOCK OB1
TITLE =
VERSION : 0.1
VAR_TEMP
    OB1_EV_CLASS : BYTE ;    //Bits 0-3 = 1 (Coming event), Bits 4-7 = 1 (Event class 1)
    OB1_SCAN_1 : BYTE ;      //1 (Cold restart scan 1 of OB 1), 3 (Scan 2-n of OB 1)
    OB1_PRIORITY : BYTE ;    //1 (Priority of 1 is lowest)
    OB1_OB_NUMBR : BYTE ;    //1 (Organization block 1, OB1)
    OB1_RESERVED_1 : BYTE ;  //Reserved for system
    OB1_RESERVED_2 : BYTE ;  //Reserved for system
    OB1_PREV_CYCLE : INT ;    //Cycle time of previous OB1 scan (milliseconds)
    OB1_MIN_CYCLE : INT ;     //Minimum cycle time of OB1 (milliseconds)
    OB1_MAX_CYCLE : INT ;     //Maximum cycle time of OB1 (milliseconds)
    OB1_DATE_TIME : DATE_AND_TIME ; //Date and time OB1 started
END_VAR
BEGIN
NETWORK
TITLE =

    CALL FB 84 , DB 84 (
        IN_DT := #OB1_DATE_TIME);

END_ORGANIZATION_BLOCK
```